WHAT FEEDBACK DOES A PROMPT-OPTIMIZED LOGIC TRANSLATOR NEED UNDER A PROOF-BASED JUDGE?

Anonymous authors
Paper under double-blind review

ABSTRACT

Translating natural language into formal logic lets a theorem prover do the inference: reasoning becomes a checkable procedure, and an entailment verdict comes with a proof. What such a translator needs is rarely measured, because published systems are evaluated as packages, each under its own data, models, and judge. We hold the judge fixed, one proof-based judge for every condition, and vary the feedback a prompt-optimized LLM translator receives, its weights frozen. Several models are evaluated on lexical, structural, and monotonicity entailment datasets. We compare feedback sources (target formulas from a symbolic pipeline or entailment labels with the prover's diagnostics), feedback channels, delegation to notation or code, and sentence-level versus pair-level translation. With a strong reflection LM, label feedback fixes the conventions the training pool penalizes, but leaves the two calls disagreeing on argument structure and on predicate names: under a matched training pool, the label-trained translator trails the formula-trained one by eight points on lexical entailment. The condition that shows the formulas as text in the feedback closes that gap to the one that scores against them. A code layer or a one-page notation in the seed prompt each make up a large part of the shortfall without the formulas. Applied to the pair instead, the same prover feedback scores highest on lexical entailment and lowest on the non-entailment list; but on lexical entailment two of every five pairs it proves are merged, the hypothesis formula mapping into the premise formula although the hypothesis sentence has a word the premise lacks.

1 INTRODUCTION

Translating a sentence into a logical formula and handing inference to a theorem prover gives something a classifier cannot: an answer with a proof. When the prover says *entailed*, a proof stands behind the verdict; when no proof is found, the system declines to answer. Such translators have been built in three main ways. Symbolic pipelines parse the sentence with a grammar and build the formula from semantic rules, and both need experts to write them (Mineshima et al., 2015; Martínez-Gómez et al., 2016; Hu et al., 2020). Trained translators put the mapping into weights, learning from gold sentence–formula pairs and verifier rewards (Yang et al., 2024b), from a compiler and consistency reward without labels (Huang et al., 2025), or from responses ranked by a theorem prover (Xu et al., 2026); they need an annotated corpus in the target notation, or a training run per notation. And LLM-based translators are controlled by a prompt, with no access to model weights: written by hand (Olausson et al., 2023; Pan et al., 2023; Ye et al., 2023), repaired at test time against solver feedback (Quan et al., 2024; 2025), or optimized against it, so far for auxiliary modules rather than for the translator itself (Liu et al., 2026b).

Our focus is the last of these. We fix the translator as a black box (a frozen LLM changed only through its prompt) and ask which feedback makes it a good logic translator. The published record is hard to read for this purpose, not because it is thin but because each system bundles its choices. The feedback a translator learns from, the notation it emits, the judge that scores it, and whether premise and hypothesis are translated together or apart all differ from one paper to the next, so a comparison across papers changes several of them at once, and the effect of any single choice cannot be read off. A practitioner starting a new domain must make the same choices, and the literature offers little guidance on any one of them alone. Our contribution is an empirical study that

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.

separates the roles of the supervision, its channel, delegation to notation or code, and the translation unit in prompt-optimized logic translation, under one judge.

One design choice shapes all the experiments: the premise and the hypothesis are translated in *separate calls*, and neither call sees the other sentence. A premise translated once must later match hypotheses that did not exist when it was translated, and separate calls also rule out shortcuts that work only for one specific pair. With a proof-based judge this has a direct consequence. The prover can connect two formulas only if they use the same predicate names, the same argument structure, and the same modeling choices. If the two calls disagree on any of these, the goal is unprovable even when each formula is a good translation on its own. A pair can therefore fail for two separate reasons: a formula that misstates its sentence, or two formulas that state it in incompatible ways.

Our experiments yield four main findings. First, label feedback and prover diagnostics help establish structural conventions exposed by the training pool, but separately translated sentences can still disagree on argument structure and predicate names (§3). Second, target formulas improve this coordination through textual feedback: showing them to the optimizer yields performance comparable to scoring against them, while a one-page notation guide or a short code adapter recovers much of the benefit without target formulas (§4). Third, pair-level refinement achieves the highest SICK accuracy, but can rewrite formulas to support both valid and invalid entailments, highlighting the gap between task accuracy and translation faithfulness (§5). Fourth, the optimized prompts encode mappings from syntactic configurations to formulas, sometimes applying a uniform rule where the appropriate interpretation depends on the lexical item (§6). We then probe the sensitivity of these findings to the optimizer, reflection LM, task model, and training distribution (§7).

2 Setup: One Judge, Six Conditions

Optimizer. All translators are produced by GEPA (Agrawal et al., 2026), a reflective prompt-evolution procedure, in its DSPy implementation (Khattab et al., 2024). The task model's weights are frozen; what evolves is the instruction text. In each step a reflection LM reads a minibatch of per-example results (the score and a textual feedback string) and proposes a revised instruction, and candidates survive by their score on a held-out validation pool. Two channels therefore enter the optimizer: a scalar reward that selects among candidates, and the feedback text that the reflection LM conditions on when writing the next candidate. The reflection LM is fixed across every condition (Claude Opus 4.8) and treated as part of the optimizer; we vary it once (§7).

Task models. The open-weight task models are Qwen-235B and Qwen-32B, a within-family scaling pair; Llama-70B, a cross-family control; and Qwen-Coder-30B, a code-specialized contrast. Qwen-235B is used for every analysis and Qwen-32B for replication; one frontier model (GPT-5.6-terra) is run once per arm (L and S). Native thinking is disabled, so all reasoning happens in the visible chain-of-thought field the prompt controls (Appendix A.1); all models are accessed through one API gateway at temperature 0.

Feedback arms. Table 1 lists the six optimized conditions. Three of them are the feedback arms of §3, identical in budget and differing in what feedback they receive. *The silver arm (S, silver reward)* scores each generated formula by prover-checked equivalence with a *silver formula* (a graded score: 0.2 each for well-formedness, each entailment direction between the two formulas, equivalence, and exact match; since equivalence follows from the two directions, a formula that means the same as the silver formula scores 0.8 before exact match and one that only entails it 0.4, a weighting chosen to reward faithful translation over partial overlap) and shows the silver formula in feedback. The silver formulas are not hand-written: they are the outputs of the symbolic pipeline of Appendix B on the pairs where that pipeline's own verdict matched the entailment label (hence silver, not gold), in the pipeline's event-semantics notation (Davidson, 1967; Parsons, 1990) (example in Appendix A.1). *The label arm (L, label reward)* is label-only with prover-derived diagnostics: it never sees a silver formula; the pair's two formulas are judged by the prover, the reward is 1.0 when the verdict equals the label (on SICK's contradiction pairs an unknown verdict also scores 1.0), 0.2 for a wrong label with well-formed formulas, and 0 for a parse failure, and the feedback shows the model's own formulas, the prover's verdict, and the prover's diagnostics (the stuck subgoal, hypothesis predicates missing from the premise, and any WordNet axioms injected (Miller, 1995)). *The bare arm (L-bare)*

Table 1: The optimized conditions. Every condition uses the same optimizer, task model, reflection LM, and judge; the label reward is 1.0 for a correct verdict. Feedback text is what the reflection LM reads beside the score. Seeds 0, 1, 2 of every condition are optimizer seeds; the single-run probes of §7 are listed in Appendix A.1.

Condition	Training data	Reward	Feedback text	§
L (label arm)	full pool, 16,734 pairs	label	pair verdict, own formulas, prover diagnostics	3
L-bare (bare arm; no WordNet axioms in reward or judging)	full pool, 16,734 pairs	label	as L, without axiom traces	3
S (silver arm)	silver pool, 6,420 sentences	silver equivalence	silver formula, score	3, 4
control	silver pool, 3,117 pairs	label	as L	4
silver-view	silver pool, 3,117 pairs	label	as L, plus the silver formulas	4
notation (notation page in the seed prompt)	silver pool, 3,117 pairs	label	as L	4

removes the WordNet abduction step from both reward and judging. The coarse $0/0.2/1.0$ reward is deliberate: everything about *what* to fix comes from the feedback text.

Training pools. The label arms optimize on pairs drawn from the SICK and HELP training splits and from HANS, which has no official training split, so its pool is cut disjointly from the 1,050-pair HANS test list; overlaps with the auxiliary lists are reported in Appendices A.6 and B (16,734 training pairs: SICK 4,734 / HANS 6,000 / HELP 6,000; GEPA scores candidates on a fixed validation subsample of 100 pairs, stratified by source). The silver arm optimizes on the silver-formula mix of the same sources (6,420 sentences, balanced; 150-sentence validation subsample). Both pools contain HELP in similar proportion (36% and 33%), and the label pools contain 9,593 pairs whose gold label is not entailment (HANS 3,200, HELP 3,010, SICK 2,738 neutral and 645 contradiction), so the reward penalizes a false proof where shallow heuristics fail.

Evaluation datasets. We evaluate on three test lists: SICK (Marelli et al., 2014b) (2,116 pairs; lexical entailment; entailment 1,404 / contradiction 712, neutral pairs excluded), HANS (McCoy et al., 2019) (1,050 pairs; structural entailment; 70 pairs from each of the 15 entailment subcases), and MED (Yanaka et al., 2019b) (1,000 pairs; monotonicity; entailment 506 / non-entailment 494). The 570 HANS non-entailment pairs (all gold unknown) are a separate list for counting false decisive verdicts (Table 6). All training splits are disjoint from the three test lists. SICK and HANS are in-distribution held-out lists, so they measure whether the constructions of the training pool are handled consistently rather than robustness to heuristics. No arm trains on MED, but both pools contain the automatically generated HELP (Yanaka et al., 2019a), which covers the same monotonicity phenomena, in similar proportion, so MED is a held-out benchmark rather than a zero-shot one.

Judge and scoring. The judge is one Rocq (formerly Coq) pipeline (Bertot & Castéran, 2004) shared by every condition: prove the hypothesis, else its negation, else return *unknown*. It proves either notation (Appendix A.2). Scoring is *strict* on every list, meaning the judged label must equal the gold label. On MED, where half the pairs are gold unknown, accuracy is read together with the false-decisive rate (a translator that proves nothing scores .494).

Budgets and reporting. All optimizations in Table 2 receive identical budgets (4,000 task-LM calls; the probes of §7 are budgeted in Appendix A.1). The reflection LM is called 41–59 times in the label arms and 25–26 times in the silver arm (Appendix A.5). Cells with three seeds show mean$\pm$std over three optimization runs (seeds 0, 1, 2; "seed" alone means the optimizer's random seed, never the seed prompt), single runs are tagged with their seed, and significance tests compare the per-pair records of two specific runs. The budget is matched, not a convergence point (Appendix A.5).

3 How Far Optimization Takes a Prompt

This section measures what optimization adds to a handwritten prompt, taking each kind of feedback as far as its own data carries it at a matched budget. Entailment labels exist for every pair of the training sources. Silver formulas exist only where the pipeline's verdict matched the label (§2), and such a filter does not admit neutral pairs: a yes or no verdict is backed by a proof, which checks the two formulas against each other though not against their sentences, whereas an unknown verdict is backed by nothing, and a formula that misstates its sentence yields it as readily as a correct one. The silver pool is therefore a subset the pipeline selects and an easier one, with no neutral pairs.

Table 2: The feedback conditions (mean±std over three seeds where shown; handwritten-prompt rows are single runs), each trained on the pool its feedback supplies (Table 1). Same-pool comparison: Table 3. Full grid: Appendix C; single-run probes: Table 13.

Model	Condition	SICK	HANS	MED
Qwen-235B	neutral prompt	.166	.290	.522
	notation prompt	.322	.615	.554
	L-bare	$.247 \pm .034$	$.892 \pm .017$	$.557 \pm .042$
	L	$.368 \pm .008$	$.909 \pm .016$	$.593 \pm .022$
	S (silver)	$.495 \pm .022$	$.901 \pm .015$	$.675 \pm .003$
Qwen-32B	neutral prompt	.090	.098	.515
	notation prompt	.292	.633	.538
	L-bare	$.223 \pm .038$	$.864 \pm .041$	$.517 \pm .006$
	L	$.336 \pm .026$	$.848 \pm .032$	$.569 \pm .021$
	S (silver)	$.438 \pm .008$	$.750 \pm .051$	$.617 \pm .024$

Table 2 has two handwritten baselines. The *neutral prompt* is a minimal description of the task; the *notation prompt* additionally describes the event-semantics notation of §2. Optimization improves on both by a large margin: from the neutral prompt, label-only optimization raises SICK from .166 to .368 and HANS from .290 to .909 at 235B.

Removing WordNet abduction (L-bare against L) costs .121 strict on SICK at 235B, .036 on MED, and nothing detectable on HANS (L-bare is .017 lower, within seed noise): the word-level relations that SICK entailments turn on (a man is a person) are supplied by WordNet at judging time, and without it the prompt does not supply them. The MED column shows the same ordering with smaller margins, on the held-out benchmark. The one reversal, silver below L on HANS at 32B, is a parse-failure cost: the silver arm fails to parse more HANS pairs than L at every seed, and its accuracy falls with the failure count (Table 7 in Appendix C). Where the silver arm's margin lies is the subject of §4; §6 reads the learned construction policies and shows where they stop.

Two reference systems place the absolute level (Appendix B): a hand-built symbolic pipeline with WordNet and a ByT5 parser fine-tuned on the silver formulas, judged by our judge on the same pairs as the label arm. On SICK and MED the label arm scores below both and on HANS above both, with complementary errors. Prompts alone, then, reach a translator that scores above two resource-heavy systems on structural conventions and below them on lexical entailment; the systems that hold a lexicon, in a grammar or in weights, win where the lexicon decides.

4 What Silver-Formula Feedback Adds, and How Much of It Can Be Recovered Without It

§3 measured what optimization adds on the data each feedback comes with; here the data is held fixed. Every condition in this section trains on one pool, the silver pool, and is compared with one baseline, the label reward on that pool (*control*). The silver arm trains on the pool's sentence records; the pair conditions train on its complete pairs, and the 186 SICK pairs split across the training and validation records are rejoined into the validation pool. The pairs are entailment-heavy for the reason given in §3: 2,738 entailments (HANS 1,070, HELP 1,070, SICK 598) and 379 SICK contradictions, no neutral pairs, so the label reward here penalizes a false proof on those 379 pairs only, against 9,593 in the full pool; that is why the control scores above the label arm of §3 on SICK (.415 against .368). On this matched pool (Table 3, first and third rows), the silver reward's margin over the label reward is eight points on lexical SICK, six on held-out MED, and within seed spread on structural HANS. Silver-trained translations also stay closest to the words of their source sentence: the share of a sentence's content words missing from its formula is about half the label arm's on SICK and MED, and the label and bare arms are level (Appendix A.7). What the silver formulas add on these lists is agreement between the two formulas rather than more lexical bridging: the arity conflicts and predicate-name variants that separate the label arm's two calls, over a quarter of its SICK errors (label arm of §3, Appendix E), fall under the silver reward, the arity conflicts to almost none. We call the rules behind the name variants *naming conventions*: how a content word becomes a predicate symbol (lemma form, number, compounds, numerals, proper names). The rest

Table 3: The matched-pool conditions (235B; mean±std over three seeds). All train on the silver pool; the last row is the control's outputs judged after the adapter (Appendix A.3).

Condition (same data pool)	SICK	HANS	MED
label reward, no silver formulas (control)	$.415 \pm .027$	$.877 \pm .028$	$.613 \pm .041$
label reward + silver formulas in feedback (view)	$.503 \pm .034$	$.903 \pm .022$	$\mathbf{.677 \pm .009}$
silver-equivalence reward (silver arm)	$.495 \pm .022$	$.901 \pm .015$	$.675 \pm .003$
label reward, notation written into the seed prompt (notation)	$\mathbf{.511 \pm .025}$	$.919 \pm .006$	$.644 \pm .048$
label reward, judged after the adapter (control + adapter)	$.451 \pm .035$	$\mathbf{.946 \pm .019}$	$.615 \pm .050$

of this section asks through which channel that value arrives, and how much of it two substitutes recover: a written notation and a code layer. Every comparison below is over three seeds; the seed-paired tests and the per-seed counts behind each statement are collected in Appendix A.7.

Reward or text. The silver arm receives the silver formulas through two channels at once: as the reward, which scores equivalence with them, and as text, since its feedback shows them; its prompt ends up producing their event notation. The *silver-view* condition keeps only the text channel. It uses the L label reward and appends the failing pair's silver formulas to the reflection feedback; a matched *control* trains on the same pool with the same reward and no formulas. The text carries the notation. The silver-view prompt contains a section that restates the event notation's rules, nearly every silver-view output uses the event notation, and no label-only arm uses its role functions (§6). Silver-view beats its control on SICK and MED in every seed and is level with it on HANS in the mean (one seed below, $p = .046$), and it shows no consistent difference from the silver arm across seeds (Table 3). The condition that shows the formulas as text in the feedback therefore closes the gap to the one that scores against them. Silver-view also pays the cost of the notation. Its parse failures on SICK rise to the silver arm's level. The reason is the same in both arms: their formulas nest more deeply than the label arm's (mean depth 5.2 against 2.2 at seed 0; 3.0–5.4 across silver-view seeds), and long sentences lose parenthesis balance. The notation alone does not force the depth: the notation condition writes the same notation at depth 3.0–3.4 (Appendix A.7).

A written notation instead of formulas. A practitioner without formulas can still write the notation down. The fourth row of Table 3 trains with the label reward on the same pool, no formula shown, and the notation written into the seed prompt as one page (event variables, role equations, noun-phrase and assembly rules, taken from the event notation). This condition is above the control on SICK in every seed and on HANS and MED in two of three; it reaches the level of silver-view and the silver arm on SICK and HANS, with no consistent direction between them, and falls short of both on MED, where one seed falls well below the other two on entailment recall ($.283$ against $.457$ and $.362$). Its outputs use the event notation on every sentence with a verb event (copular sentences such as *John is a student* have none), and it inherits the notation's HANS trade-off in every seed (understood objects and passives solved, complement clauses under a verb lost) but not its parse-failure cost: its formulas nest between the label arm's and the silver arm's, and its parse failures on SICK (20–36 pairs) sit between the label arm's (4–8) and silver-view's (47–160). The written page leaves one gap the reward does not close: in two of three seeds most of its false decisive verdicts on SICK (Appendix A.7) come from one convention its prompts fix, a plural subject rendered as two distinct entities that are both the subject of one event, which the prover reads as a contradiction.

A code layer instead of the prompt. On HANS, the error class that dominates the label arms is the arity conflict (§6): the two calls give the same verb a different number of arguments, and the prover cannot connect them. A code layer (the *adapter*, Appendix A.3) applied to both formulas before proving rewrites every predicate with two or more arguments as a unary event predicate with argument links, which makes arity irrelevant and restores the one-place shape WordNet axioms need. Applied to the control's outputs (Table 3), it recovers 66–86 HANS pairs per seed with at most one regression, lifts SICK by three to five points (McNemar $p < 10^{-11}$ in every seed), and leaves MED unchanged. The same pass applied to the label arm gives the same gains (Appendix A.7) and, on the non-entailment list, adds a few false proofs (Table 6).

What the substitutes cover. Take the distance from the control to the silver arm on the matched pool as 100%. On SICK the adapter closes 45% of it, and silver-view and the written notation reach

Table 4: Test-time refinement against optimized prompts (235B; refinement rows are single runs). Accuracy columns: three-seed means for L. HANS non-ent.: the 570-pair non-entailment list (gold unknown, Table 6). Merged: share of SICK pairs judged entailed that are merged by the definition of §5; P=H: the share whose two formulas are identical strings (seed-0 run for L). Last column: mean LLM calls per pair on SICK / HANS / MED.

Method	SICK	HANS	HANS non-ent.	MED	merged (SICK)	P=H (SICK)	calls/pair
L (optimized)	$.368 \pm .008$	$.909 \pm .016$	$\mathbf{.758 \pm .009}$	$.593 \pm .022$	8%	26%	2
refinement from the neutral prompt	.716	.819	.746	.573	43%	46%	3.5 / 3.3 / 4.7
refinement from L	**.805**	**.982**	.665	**.693**	41%	40%	3.1 / 2.1 / 3.9

or exceed the silver arm's score. On MED the adapter closes none of it, the written notation half, and silver-view all of it. On HANS the conditions are within seed spread of one another and the distance (.024) is too small to divide. What the adapter recovers is the arity convention and nothing of the naming conventions. A practitioner with labels and no formulas can therefore add the adapter, or the written notation where a notation can be written down.

5 One Sentence or the Pair

So far the translator has seen one sentence at a time: the prover's feedback is computed on the pair, but it is accumulated into a prompt that translates each sentence alone. The feedback can instead act on the pair at test time, and this is the main way prior LLM translators use it: a handwritten prompt and a refinement loop that shows the solver's feedback on the pair back to the model (Pan et al., 2023; Quan et al., 2024). Our refinement baseline translates both sentences with the neutral prompt and, whenever a formula is ill-formed or the prover cannot connect the pair, shows the model *both* sentences, *both* formulas, and the same diagnostics GEPA's reflection saw, asking for jointly revised formulas (up to three rounds; no labels are used).

On SICK, refinement beats the optimized prompt (.716 against .368), and its outputs show what the margin is made of. In 46% of the pairs it proves (533 of the 1,151 it judges entailed), the revised premise and hypothesis formulas are identical, an identity made by the refinement, not found in the sentences. It copies one sentence's wording onto the other (*ball* becomes *toy*, *boy* becomes *child*), and the prover then proves a formula against itself. Identity is the limit of a more general operation. We call a proved pair *merged* when the hypothesis formula maps into the premise formula (every atom, with its predicate, arity, and argument positions, under one renaming of variables) although the hypothesis sentence has a content word the premise lacks: the lexical gap was closed by rewriting a formula rather than by an axiom. The merge share is a diagnostic, not a direct measure of unfaithfulness: a legitimate lexical normalization satisfies the same criterion. A mapping alone is the ordinary proof route (63% of L's SICK proofs, 99% on HANS); the merge measure separates the systems: 43% of refinement's SICK proofs and 33% of its MED proofs, against 8% and 14% for L and 5% and 4% for the silver arm (Appendix A.4). The optimized prompt produces identical pairs in 26% of its proofs, mostly for pairs whose sentences already share their words: only 12% of them have a hypothesis word the premise sentence lacks, against 54% of refinement's. From the neutral prompt the merges cost few wrong verdicts on the three main test lists (13 false decisives against L's 10 on SICK; 43 against 67 on MED), though five of the SICK ones are antonymic pairs merged into one formula (*hanging up* the phone and *making a call*); what they cost where every new proof is wrong is shown below. One further property follows from repairing one pair at a time: the output format degrades. Refinement's ill-formed pair counts are 266 / 63 / 411 on SICK / HANS / MED against L's 8 / 2 / 36. On structural data the loop is below the optimized prompts (Table 4).

The last row of Table 4 runs the same refinement loop from L's seed-0 formulas, on the pairs the prover left unknown. The same merges appear (41% of its SICK proofs), and what the pair view does is the same on every list: the revision rewrites one formula until the hypothesis maps into the premise, creating that mapping in 93% of the SICK pairs it flips to yes, 81% on HANS, 59% on MED, and 95% on the HANS non-entailment list (Appendix A.4). On HANS the words are shared by construction, so the rewrite is structural: it changes argument structure, on the non-entailment list by editing the premise alone in 41 of 59 pairs, collapsing a clause the premise does not assert into an object (*believed the tourist thanked the bankers* becomes *believed the tourist*), the treatment of clausal complements that §6 finds fixed in the prompts. The same operation is a repair on the

entailment list (88 of 88 flipped pairs correct) and a fabrication on the non-entailment list (59 of 59 wrong), and the loop, which sees no label, cannot tell them apart. On that list, where the correct verdict is always unknown, refinement from L falls nine points below L's mean and refinement from the neutral prompt one point (Table 4; the two HANS lists averaged with equal weight give .833 for L and .824 after refinement from L), and on MED the wrong decisive verdicts double (67 to 146), outweighed there by 178 correct flips. The .805 on SICK, like refinement's .716, is therefore a decision accuracy rather than a translation accuracy.

Under either reading the starting point matters. Refinement from L lifts SICK from .372 to .805, HANS from .898 to .982, and MED from .594 to .693, and on the three main test lists the cell is above refinement from the neutral prompt, so the conventions accumulated in the prompt and the view of both sentences each contribute; on the non-entailment list it is below it (.665 against .746), so the optimized start proves more pairs, with fewer ill-formed formulas than the same refinement from the neutral prompt (Table 4; ill-formed counts in Table 5 of Appendix A.4), and also fabricates more.

6 Reading the Errors and the Prompts

Two things are readable after optimization: the wrong pairs and the prompts. Every wrong pair is assigned to one of five classes: parse failure, arity conflict, predicate-name variant, lexical gap, or structural (Appendix E); Figure 3 in that appendix plots the 235B cells. Under the label reward at 235B, optimization reduces parse failures and name variants on every set; the silver-fed conditions reduce name variants but leave more parse failures on SICK than the label arms (§4). What remains under the label reward is lexical gaps on SICK and MED and structural errors on HANS, where arity conflicts are 58% of L's errors and 28% of L-bare's at 235B. The errors are conservative (false decisive counts in Appendix C).

The prompts show two regularities. First, what reaches the prompt is what the feedback shows: the event notation appears only where silver formulas or their notation page were shown. Second, what accumulates is procedure, not knowledge of the phenomena the data test: across the GEPA-evolved prompts, monotonicity vocabulary occurs zero times and no prompt distinguishes factive from non-factive verbs; the rules concern quantifier templates, indexicals, negation scope, and argument order.

Three HANS constructions separate the cells (Table 8 in Appendix D pairs each with the rule every prompt states for it). Two of them are decided by one sentence of the prompt. Understood objects ("The actor paid the lawyer" $\models$ "The actor paid") need a valency policy in the flat notation: prompts that read arity off the sentence lose the construction (.00–.14), the one prompt that reads it off the verb recovers most of it (.69), and the event notation and the adapter handle it without any rule (.99–1.00). Passives: eleven of the twelve seed-0 prompts of the grid (Table 7) state an argument policy fixed by thematic role that ignores voice (.74–1.00); the one with a surface convention scores .00. Complement-clause entailment ("The president knew that the doctors resigned" $\models$ "The doctors resigned") depends on the factivity of the verb, and the training pool separates the cases (210 factive entailments beside 201 non-factive non-entailments), yet no prompt states a rule that depends on the verb: each applies one policy to every clausal complement, stated in two prompts and implicit in the rest (Table 8), and the accuracies (.14–.94) record it. The non-entailment list shows the cost of that policy (Table 6 in Appendix A.6): every optimized 235B condition, the silver arm included, returns a decisive verdict on 16–25% of its pairs, and 89–94% of those wrong proofs fall in four constructions, the non-factive complements among them. What the label signal fixes is a mapping from syntactic configuration to formula; a property that varies with the word, like factivity, gets one policy for the whole configuration, although the pool penalizes that policy on half of its cases.

Table 9 in Appendix D reads the six label-only prompts of the 235B and 32B seed triples along ten representational axes. Four hold in every run: the base grammar (two axes), arity, and voice; one, the base-form lemma, holds in all but one run. The pool penalizes each of the last three: passives by voice, arity by understood objects, the lemma by pairs that use the same verb in different forms. Four vary across seeds of the same model, sometimes in opposite directions; no training pair separates any of them, and without one no convention became fixed. With one, the right convention is not guaranteed: the understood object is penalized by the pool, yet every L prompt that states a rule reads arity off the sentence and loses the construction (the valency rule that recovers it, .69, is the bare arm's). Passives are 2–3% of both pools, and twelve of the fifteen label-reward runs (L, L-bare,

control, silver-view, notation) settled the thematic convention (passive recall $\geq .90$); the exceptions are control seed 0, whose prompt has no passive rule (.70), control seed 2, which fixes surface argument order (.49), and silver-view seed 1, which kept a surface convention (`_by(e, x)`) and lost the construction (.39) although its feedback showed the thematic form on every passive pair. The open axes are the conventions a formula design will have to fix in advance, in the seed prompt or by a rewrite like the adapter; the fixed ones can be left to the optimizer.

7 How Far the Answers Depend on the Recipe

The sections above hold the optimizer, the reflection LM, the task-model family, and the data fixed; the probes below vary one of these each, one run per condition, so we read directions, not sizes. The cells are in Table 13 (optimizer, reflection LM, task model) and Tables 14 and 15 (distribution).

Optimizer. MIPRO (Opsahl-Ong et al., 2024) searches over instructions and bootstrapped demonstrations under the same L reward, using the scalar score alone, with no feedback text. At a budget matching GEPA's it stays below the notation prompt on SICK, below every GEPA L run on SICK and HANS (McNemar $p < .001$), and below all three on MED, two of them significantly. A second *reflective* optimizer, a TextGrad-style loop (Yuksekgonul et al., 2025) in our implementation, keeps one candidate: at each step the same reflection LM reads the same feedback text for a minibatch, writes a critique of the instruction, rewrites it, and the rewrite is kept if the validation score does not drop. Run under the S and L rewards at the GEPA budgets and seed prompts, it reproduces the S$-$L order on SICK and MED ($p < 10^{-8}$), with margins of the size GEPA gives, and puts S ten points above L on HANS, where GEPA's two arms are level. Against the GEPA seeds its prompts fall within a point of the seed range on SICK and MED, with no consistent direction, and below it on HANS: the L prompt settled the opposite convention for bare plurals from every GEPA L prompt (universal against existential), and its fixed-arity rule loses understood objects. The direction of the feedback contrast therefore does not depend on which reflective search reads the feedback; the HANS level, which turns on a few global conventions (§6), does.

Reflection LM. The two 235B-reflection runs replace Claude Opus 4.8 with the 235B task model as the reflection LM and change nothing else, one under the L reward and one under the S reward. Both fall below their three Opus-reflection seeds on every set, the L run most on SICK (eight to nine points; McNemar $p < .001$) and the S run by ten points on SICK and eighteen on HANS. The prompt the L run wrote is half the length of the L prompts: it keeps the rules about quantifier choice and argument order and lacks the ones the L prompts carry for noun-phrase decomposition, possessives, proper nouns, and verb arity. Given the same feedback, this reflection LM dropped the naming and arity rules first. The S$-$L contrast under the weaker reflection LM keeps its direction on SICK and MED ($p \leq .046$), on MED narrower than the three-seed margins under Opus, and turns against S on HANS ($p < .001$).

Task model. At GPT-5.6-terra the S$-$L gap *widens* on SICK and MED and reopens on HANS, where the 235B gap is within seed spread. The silver arm rises with model capability on all three sets, while the label arm rises on SICK and MED but not on HANS. In this one run, capability amplified what the feedback provides rather than substituting for it.

Distribution. Under the test shift, the evolved prompts applied as they are to FOLIO (Han et al., 2024) (multi-premise) and FraCaS (Cooper et al., 1996) score within four points of the all-unknown baseline on FOLIO (.340) and, at 235B, nine to fourteen points above it on FraCaS (.293), except the silver arm, which transfers worst on FraCaS. Its poor transfer is not caused by its notation: silver-view and the written notation emit the same event notation and transfer at L's level, proving as many pairs as L does, whereas the silver arm proves almost none. The pair-level refinement of §5 applied on these lists raises every cell it is run on, most on FraCaS's multi-premise half (Appendix F). Under the training shift, re-optimizing on the new data with the evolved prompt as the starting point gives opposite answers on our two testbeds. On MED (single-premise, close to the training distribution) the evolved start beats a neutral restart (McNemar $p < 10^{-4}$), with fewer false claims. On FOLIO (multi-premise, a structural shift) the evolved start is worse in this single run (13 against 4 discordant pairs, $p = .049$). On these two testbeds, whether an evolved prompt is worth restarting from therefore depends on the distance to the new distribution.

8 Related Work

Learning to map language to formal representations from a downstream signal rather than gold logical forms is a long-standing approach (Clarke et al., 2010; Liang et al., 2013; Berant et al., 2013; Artzi & Zettlemoyer, 2013; Goldman et al., 2018); our label-only arms are this approach with a frozen LLM as parser and a prover's verdict as signal. Recognizing entailment by translating to logic and calling a prover goes back to Bos & Markert (2005); symbolic NLI pipelines in this line build formulas from parses with hand-written rules and fill lexical gaps by axiom injection or by phrase correspondences acquired from proofs (Mineshima et al., 2015; Martínez-Gómez et al., 2016; Abzianidze, 2017; Yanaka et al., 2018; Hu et al., 2020); our judge, silver formulas, and abduction mechanism come from this line. LLM-based systems translate into formal languages and delegate inference to solvers (Olausson et al., 2023; Pan et al., 2023; Ye et al., 2023).

The closest work on the feedback axis trains weights. LogicLLaMA uses an FOL verifier as an RLHF reward (Yang et al., 2024b). FormaRL trains an autoformalizer with GRPO on a binary reward from a Lean syntax check and an LLM consistency check (Huang et al., 2025). LogicReward scores reasoning steps with Isabelle, repairs them from its error messages, and trains on the best- and worst-scored responses with SFT and DPO (Xu et al., 2026). All three change the weights; we keep the weights frozen and route the prover's verdict and diagnostics into the text of a prompt. ToMap evolves an auxiliary decomposer inside a Lean autoformalization pipeline (Liu et al., 2026b); the translator itself is not optimized. Among automatic prompt optimizers (Zhou et al., 2023; Pryzant et al., 2023; Yang et al., 2024a), we build on GEPA (Agrawal et al., 2026) in DSPy (Khattab et al., 2024), with MIPRO (Opsahl-Ong et al., 2024) as the scalar-only comparison and a TextGrad-style loop (Yuksekgonul et al., 2025) as a second reflective search; the contribution is not an optimizer but a measurement of which parts of the feedback it turns into capability.

Quan et al. verify NLI explanations in Isabelle/HOL and repair them from failed proof steps (Quan et al., 2024; 2025); the feedback is consumed per instance and discarded, which is the unit question of §5. Kim et al. (2026) test whether fixed translators game a Lean checker, and Amrollahi et al. (2026) enforce faithfulness by roundtrip verification with repair; our surface-departure and false-decisive measurements ask instead which feedback signal produces faithful translations.

9 Conclusion

Under a fixed proof-based judge and with a strong reflection LM, prover-derived label feedback is sufficient to recover many structural representation conventions, while silver-formula feedback mainly adds the event notation and agreement in argument structure between the two calls, and part of the naming conventions; the condition that shows the formulas as text in the feedback closed the gap to the one that scores against them. Without formulas, a one-page notation in the seed prompt and a code layer each make up a large part of the shortfall (§4). The same prover feedback applied to the pair instead, with both sentences in view and both formulas revised together, scores higher on lexically dominated data; but there two of every five pairs it proves are merged, and the same rewriting that repairs an entailment fabricates one where none holds (§5). What a prompt accumulates is procedure, rules from syntactic configuration to formula, and in the runs examined a penalizing pair in the pool was necessary for a convention to become fixed but not sufficient (§6).

Limitations. Apart from the TextGrad-style and 235B-reflection probes of §7, every result comes from one reflective optimizer with one reflection LM; what label feedback recovers here is what it recovers through a reflective feedback channel (the scalar-only optimizer recovers much less), and the absolute levels depend on the reflection LM, one run per arm (§7). Sensitivity of GEPA to its seed and reflection LM has been reported on mathematical reasoning (Liu et al., 2026a); our control runs, whose prompts fixed different surface conventions at different costs (§6), are an instance. All findings concern English data. What governs transfer we could not analyze: the transfer and re-optimization testbeds differ from the training pool in premise count (FOLIO, FraCaS), vocabulary, and construction mix at once, and the transfer probes are one run per condition. Separating these needs target distributions that vary one property at a time.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods (the optimization, evaluation, and analysis scripts), to design and give feedback on experiments, and to interpret results (statistical tests and error analyses). We have not used generative AI tools to generate synthetic data, to formulate or prove mathematical claims, or to propose the hypotheses of this work; translation, data cleaning of third-party datasets, and qualitative thematic analysis are not applicable. Additionally, we used generative AI tools to draft and edit parts of the paper, to create figures, and to identify and format references. We have reviewed all AI-assisted work: all code was run end to end by the authors, every reported number was recomputed from the stored records, and every claim in the text was checked against the underlying data by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We will release, upon publication, the optimization and evaluation code, the frozen test lists, every evolved instruction, all per-pair verdict records, and the judge scripts. Given the stored translations, every number in this paper is recomputable locally and deterministically; no API access is needed to verify any judgment.

REFERENCES

Lasha Abzianidze. LangPro: Natural language theorem prover. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 115–120, 2017. doi: 10.18653/v1/D17-2020.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Daneshvar Amrollahi, Jerry Lopez, and Clark Barrett. Faithful autoformalization via roundtrip verification and repair. *arXiv preprint arXiv:2604.25031*, 2026.

Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions*. Springer, 2004.

Johan Bos and Katja Markert. Recognising textual entailment with logical inference. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pp. 628–635, 2005. URL https://aclanthology.org/H05-1079.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL)*, 2010.

Robin Cooper, Dick Crouch, Jan van Eijck, Chris Fox, Josef van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, and Steve Pulman. Using the framework. Technical Report LRE 62-051 D-16, The FraCaS Consortium, 1996.

Donald Davidson. The logical form of action sentences. In Nicholas Rescher (ed.), *The Logic of Decision and Action*, pp. 81–95. University of Pittsburgh Press, Pittsburgh, 1967.

Omer Goldman, Veronica Latcinnik, Ehud Nave, Amir Globerson, and Jonathan Berant. Weakly-supervised semantic parsing with abstract examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander Fabbri, Wojciech Maciej Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22017–22031, 2024.

Hai Hu, Qi Chen, Kyle Richardson, Atreyee Mukherjee, Lawrence S Moss, and Sandra Kübler. MonaLog: a lightweight system for natural language inference based on monotonicity. In *Proceedings of the Society for Computation in Linguistics (SCiL)*, 2020.

Yanxing Huang, Xinling Jin, Sijie Liang, Fuwen Luo, Peng Li, and Yang Liu. FormaRL: Enhancing autoformalization with no labeled data. In *Conference on Language Modeling (COLM)*, 2025.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. In *International Conference on Learning Representations (ICLR)*, 2024.

Kyuhee Kim, Auguste Poiroux, and Antoine Bosselut. Do LLMs game formalization? Evaluating faithfulness in logical reasoning. In *VerifAI-2: The Second Workshop on AI Verification in the Wild, ICLR*, 2026. Non-archival; arXiv:2604.19459.

Percy Liang, Michael I Jordan, and Dan Klein. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446, 2013.

Shiyan Liu, Qifeng Xia, Qiyun Xia, Yisheng Liu, Xinyu Yu, and Rui Qu. Reflection in the dark: Exposing and escaping the black box in reflective prompt optimization. In *Proceedings of the ACL 2026 Student Research Workshop*, 2026a. arXiv:2603.18388.

Tian-Shuo Liu, Shiyuan Zhang, Zijie Geng, Haoyu Liu, Runjie Xu, Pengyuan Wang, Lei Yuan, and Yang Yu. Efficient test-time optimization for multi-agent proof autoformalization. *arXiv preprint arXiv:2607.11307*, 2026b.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pp. 1–8, 2014a. doi: 10.3115/v1/S14-2001.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, 2014b.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. ccg2lambda: A compositional semantics system. In *Proceedings of ACL 2016 System Demonstrations*, 2016.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

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

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

Theo X Olausson, Alex Gu, Benjamin Lipkin, Cedegao E Zhang, Armando Solar-Lezama, Joshua B Tenenbaum, and Roger Levy. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

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

Terence Parsons. *Events in the Semantics of English: A Study in Subatomic Semantics*. MIT Press, 1990.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Xin Quan, Marco Valentino, Louise A. Dennis, and André Freitas. Verification and refinement of natural language explanations through LLM-symbolic theorem proving. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Xin Quan, Marco Valentino, Louise A. Dennis, and André Freitas. Faithful and robust LLM-driven theorem proving for NLI explanations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Jundong Xu, Hao Fei, Huichi Zhou, Xin Quan, Qijun Huang, Shengqiong Wu, William Yang Wang, Mong-Li Lee, and Wynne Hsu. LogicReward: Incentivizing LLM reasoning via step-wise logical supervision. In *International Conference on Learning Representations (ICLR)*, 2026.

Hitomi Yanaka, Koji Mineshima, Pascual Martínez-Gómez, and Daisuke Bekki. Acquisition of phrase correspondences using natural deduction proofs. 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. 756–766, 2018. doi: 10.18653/v1/N18-1069.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. HELP: A dataset for identifying shortcomings of neural models in monotonicity reasoning. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM)*, 2019a.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. Can neural networks understand monotonicity reasoning? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2019b.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024a.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. Harnessing the power of large language models for natural language to first-order logic translation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6942–6959, 2024b.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. SatLM: Satisfiability-aided language models using declarative prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations (ICLR)*, 2023.

A METHOD DETAILS

A.1 OPTIMIZATION CONFIGURATION

Optimization configuration. All GEPA optimizations use dspy.GEPA with a fixed reflection LM (Claude Opus 4.8, except the 235B-reflection runs of Table 13, which use Qwen-235B at the same settings; temperature 1.0, max 32,000 tokens) and task LMs at temperature 0 (max 8,000 tokens), accessed through one API gateway. Native thinking is disabled on all task models; in a pilot, hidden thinking was ten times slower per call and no more accurate on this task. Gateway identifiers: qwen/qwen3-235b-a22b-2507, qwen/qwen3-32b, meta-llama/llama-3.3-70b-instruct, qwen/qwen3-coder-30b-a3b-instruct, openai/gpt-5.6-terra; reflection LM anthropic/claude-opus-4.8. Per-arm settings:

Arm	train pool	seed prompt	demos	minibatch	val pairs	budget
S (silver)	silver pool (6,420 sents)	none	0	5	150	4,000 calls
L / bare	pairs (16,734)	neutral (below)	0	4	100	2,000 pair-evals
control / view / notation	silver pool pairs (3,117)	neutral (notation: + notation page)	0	4	100	2,000 pair-evals
MED re-opt	MED train (932)	L evolved / neutral	0	4	120	2,000 pair-evals
FOLIO re-opt	FOLIO train	L evolved / neutral	0	2	40	600 pair-evals
TextGrad-style S / L	as S / as L	none / neutral	0	5 / 4	150 / 100	4,000 calls / 2,000 pair-evals
MIPRO (L reward)	pairs (16,734)	neutral	$\leq$ 3 bootstrapped	35	100	1,940 pair-evals

Every GEPA and TextGrad-style arm runs instruction-only: no few-shot demonstrations are attached to the prompt, so the evolved instruction text is the sole carrier of learned behavior (the reflection LM may quote training sentences inside that text, see below); MIPRO, which searches over bootstrapped demonstrations by design, is the exception. One L pair-eval translates premise and hypothesis (two task-LM calls), so label-arm budgets equal the silver arm's 4,000 task-LM calls; a FOLIO pair-eval translates every premise (5.3 on average) plus the hypothesis, so 600 pair-evals amount to $\sim$3,760 calls. Optimization seeds $\{0, 1, 2\}$ vary only `gepa_seed`; validation subsets are stratified samples with a fixed seed.

Neutral prompt (verbatim).

```
TASK: Translate a single English sentence into exactly ONE first-order
logic (FOL) formula. Output ONLY the formula in the fol field: no prose,
no labels, no code fences.
SYNTAX (the formula must parse under this grammar): connectives & | -> -
(prefix negation); quantifiers 'exists x1.' / 'all x1.' scoping over the
parenthesized body, multiple variables may share one quantifier;
variables are lower-case identifiers; predicates are lower-snake lemmas
with a leading underscore and fixed arity; equality (x1 = x2); keep
parentheses balanced; arguments are variables only.
CONSISTENCY: Formulas produced from different sentences are compared by
a theorem prover. Choose ONE systematic scheme and apply it to every
sentence: the same concept must always map to the same predicate name
and the same argument structure, so that sentences sharing content
produce formulas sharing predicates. Express only the meaning stated by
the sentence: do not add and do not drop content. Output ONLY the final
FOL formula.
```

(The released file is the source of truth; the notation prompt additionally specifies the event notation below.)

Silver formulas. The silver formulas of the silver pool are the outputs of the symbolic pipeline of Appendix B on the pairs where that pipeline's own verdict matched the gold label (6,420 sentences). Their notation is event-semantic: every verb is a one-place predicate over an event variable, and the participants are linked to the event by the role functions `Subj`, `Acc`, and `Dat`, compared by equality. For "The monkey is brushing the dog":

```
exists x1 x2 e.(_monkey(x1) & _dog(x2) & _brush(e) & (Subj(e) = x1) & (Acc(e) = x2))
```

The label-only arms settle instead on a flat notation in which the verb takes its participants directly, subject first: `exists x1 x2.(_monkey(x1) & _dog(x2) & _brush(x1, x2))`. The judge's Rocq library declares the `Event` type and the role functions, so both notations are provable under the same judge.

A.2 JUDGE

Judge. A pair is judged three-way: attempt $P_1, \ldots, P_k \vdash H$ in Rocq; on failure attempt $P_1, \ldots, P_k \vdash \neg H$; otherwise return unknown. Each attempt compiles the formulas against the ccg2lambda coqlib and runs the `nltac` tactic ladder (`Set Firstorder Depth 1. nltac. Set Firstorder Depth 6. nltac.`) under a 30-second timeout per direction. With abduction (L and all standard evaluations), an unknown verdict triggers one retry of both directions with WordNet axioms injected: for every hypothesis predicate, lexical-relation axioms (hypernymy, derivational and inflectional links) connecting premise predicates to it, generated from the formulas alone. The bare arm (L-bare) is judged without this retry, matching its own training reward.

FOL-to-Rocq builders. The legacy builder types entities and maps the flat notation's n-ary predicates directly; a second, direct builder handles constant-bearing formulas (FOLIO). On the shared notation the two builders are verdict-equivalent on our test lists.

A.3 REIFICATION ADAPTER

Reification adapter. Conflict-driven, pair-level: collect every predicate used with arity ≥ 2 anywhere in the pair (guarding predicates with propositional arguments), and rewrite each such atom $P(t_1, \ldots, t_n)$ as $\exists e\,(P(e) \wedge _arg_1(e, t_1) \wedge \cdots \wedge _arg_n(e, t_n))$ with a shared fresh-variable pool across the pair. The rewrite is applied atom by atom inside whatever scope the atom occupies, so negation and quantifier structure are unchanged, and it makes verbs one-place, so the same one-place WordNet axiom templates used everywhere else apply. Two things it assumes are stated here rather than proved. First, an occurrence of P with fewer arguments is read as the same event type with the missing arguments unspecified; this is what lets $P(x, y)$ and $P(x)$ connect, and it is a convention the code imposes, not a consequence of the flat notation. Second, the set of rewritten predicates is collected from both formulas, so the rewrite is a pair-level step. It sits between the translator and the prover: the stored premise formula is unchanged, the rewrite runs when a hypothesis arrives, and the prover, its library, and the axiom mechanism are the same as everywhere else. The separate-calls design of §1 is therefore intact; the adapter rows of Table 3 and Table 6 are the same judge applied to normalized formulas. Audit of the application (235B, seed 0) on the three entailment-bearing test lists: of 12 surface regressions, all but one are prover resource or tactic interactions, all conservative (a proof lost, not a false proof gained), and one is a semantic interaction with an already-unfaithful translation. On the non-entailment list the adapter does add false proofs (Table 6), since connecting $P(x, y)$ with $P(x)$ is exactly what its understood-object pairs require; raising the prover's budget ($4\times$ timeout, $2\times$ depth) flips no verdict, so the gains do not come from extra prover budget. This audit is evidence on the observed pairs, not a proof of meaning preservation.

A.4 REFINEMENT BASELINE

Refinement baseline. Translate both sentences with the neutral prompt; while a formula is ill-formed or the judged label is unknown, show the model both sentences, both formulas, and the same diagnostics string the optimizer's feedback uses, and request revised formulas for the pair jointly; at most three rounds; no gold labels or formulas are used. Audit of the stored outputs (235B): identical premise/hypothesis formulas (exact string match after whitespace normalization) among predicted entailments, SICK 533/1,151 (refinement) vs. 144/550 (L), 81/680 (S), 83/708 (silver-view); among these identical pairs the hypothesis sentence has a content word the premise sentence lacks in 17 of 144 (12%) for L and 289 of 533 (54%) for refinement. Among each system's own SICK proofs, the hypothesis formula maps into the premise formula in 63% (L), 62% (silver arm, silver-view), 87% (refinement), and 81% (refinement from L); merged by the definition of §5, 8% / 5% / 5% / 43% /

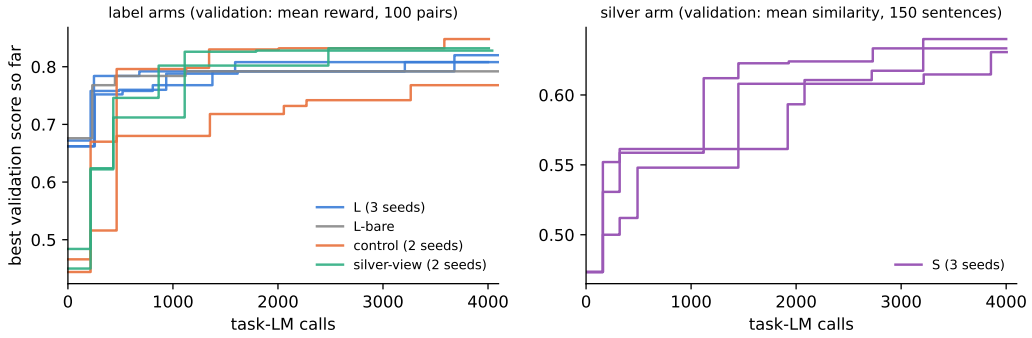


Figure 1: Best validation score so far against task-LM calls, 235B runs. Steps mark the discovery of a new best candidate.

41% (46 of 550 for L; 500 of 1,151 for refinement; 538 of 1,312 for refinement from L), and on MED 14% / 4% / 4% / 33% / 28%. The shares in §5 count, among the pairs flipped to yes, those in which the revision creates the mapping; the mapping column of Table 5 counts, among all flipped pairs, those that map after revision. Refinement from the neutral prompt on the HANS non-entailment list (one run): accuracy .746, 145 false proofs, 78 ill-formed pairs; .752 of the false proofs fall in the four constructions of Table 6. Two counts of malformed output are reported. *Ill-formed pairs* (§5, Table 5) are pairs with at least one formula the parser rejects: refinement 266 / 63 / 411 on SICK / HANS / MED, against L's 8 / 2 / 36. *Convention violations* count formulas that parse but break the output contract (unicode connectives or capitalized predicates): refinement 479 / 120 / 853 of 4,232 / 2,100 / 2,000 formulas; L 0 / 0 / 1. Rounds used on MED: 0 / 1 / 2 / 3 rounds for 44 / 94 / 23 / 839 pairs.

Table 5: What refinement changes (235B, L seed 0 before and after). Flipped: pairs moved from unknown to a decisive verdict (correct in parentheses); wrong dec.: decisive verdicts contradicting gold; P=H: share of flipped pairs with identical formulas; mapping: flipped pairs whose hypothesis formula maps into the premise formula (§5) before and after the revision; edited: which formula the revision changed (premise / hypothesis / both); ill-formed: pairs with a formula that no longer parses.

Set	pairs	accuracy	flipped	wrong dec.	P=H	mapping	edited	ill-formed
SICK	2,116	.372 → .805	927 (916)	10 → 21	42%	48 → 872	342 / 244 / 341	8 → 147
HANS	1,050	.898 → .982	88 (88)	0 → 0	10%	17 → 88	29 / 56 / 3	2 → 4
MED	1,000	.594 → .693	257 (178)	67 → 146	41%	19 → 170	52 / 77 / 128	36 → 252
HANS non-ent.	570	.768 → .665	59 (0)	132 → 191	2%	3 → 59	41 / 7 / 11	0 → 48

A.5 Learning curves and cost

Learning curves. Figure 1 plots, for the 235B runs, the best validation score reached so far against task-LM calls spent, read from the optimizer's saved state (a label-arm metric call is two task-LM calls, one per sentence; a silver-arm call is one). Most of the gain arrives in the first quarter of the budget. In the last quarter the best score still moves in nine of the eighteen runs, by fewer than four validation items, so the numbers in the paper are matched-budget numbers rather than converged ones. The two panels use different validation scores (the label reward on 100 pairs; the silver-equivalence score on 150 sentences) and are not comparable across panels.

Cost. Token counts are read from the cached API responses of every call. A reflection call (Claude Opus 4.8) averages 6,400 input and 3,900 output tokens; at list prices ($5 / $25 per million) that is $0.13 per call. A 235B task call averages 3,000 input tokens (1,300 of them cache reads) and 550 output tokens, or $0.00046 at $0.0875 / $0.35 per million. Per run at 235B: the L seeds made 52–59 reflection calls ($6.7–7.6) against about 4,100 task calls ($1.9); L-bare 41–56 ($5.2–7.2); the

control and silver-view runs 27–39 ($3.5–5.0); the silver arm 25–26 ($3.2–3.4) against 4,000 task calls ($1.8). Evaluating one prompt on the three test lists costs 8,332 task calls, about $4. The reflection LM is therefore 64–80% of an optimization run and 0.8 to 1.9 times the cost of evaluating its result; with the 235B model as its own reflection LM (Table 13) the reflection cost falls below $0.10. The TextGrad-style runs made two optimizer calls per proposal, 36 (L) and 48 (S) in all ($4.7–6.2).

Score against cost. Figure 2 replots the optimization runs of Tables 2 and 3 against cumulative cost rather than metric calls. Every API call is priced from the usage record the gateway returns with it; calls are attributed to runs by model and time window, with concurrent runs of the same model split in proportion to their known call counts, so per-run totals carry an error of some 10–20% while condition means are stable. The cost of a candidate is the task calls spent up to its discovery plus one reflection call per iteration up to its acceptance. Two things are visible. The label-reward conditions reach most of their final validation score within $2–3 and then spend the rest of the budget on small gains; the label arm's higher total ($9 against the silver arm's $6 at the same task-call budget) is reflection calls, 52–59 iterations against 25. The totals in the figure are summed from the cost field of each run's own calls, so they exceed the mean-price estimates above where a condition's calls are longer than average, as the silver arm's are. With the 235B model as its own reflection LM the L run reaches the validation level of the Opus runs at a quarter of the cost but scores lower on the test lists (Table 13); the S run stops lower on both.

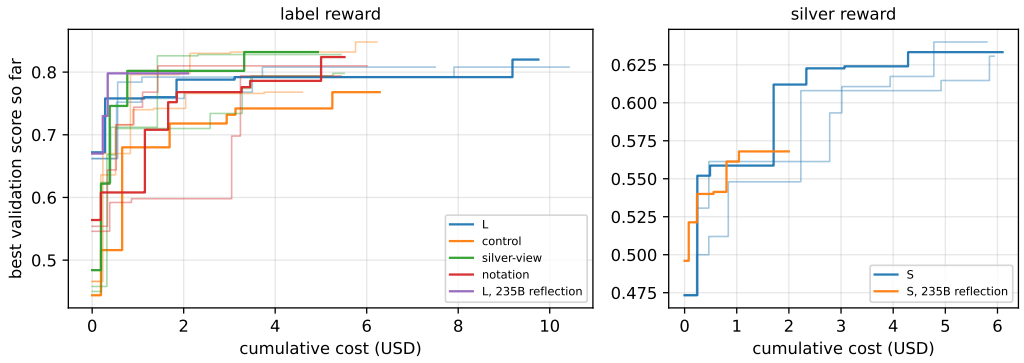


Figure 2: Best validation score so far against cumulative optimization cost (task and reflection calls, USD at list prices), 235B. Left: label reward conditions; right: silver reward. Three seeds per condition (seed 0 drawn darker); the 235B-reflection runs are single.

A.6 Test lists

Test-list construction. All three frozen lists are built deterministically from the official dataset releases by a committed script with a fixed seed. SICK is the full official TEST split minus neutral pairs (2,116 pairs: 1,404 entailment, 712 contradiction), which matches the SemEval-2014 Task 1 data statistics (Marelli et al., 2014a), with no sampling step. HANS samples 70 pairs from each of the 15 entailment-side subcases (1,050 pairs, seed-fixed within each subcase). MED first excludes the GLUE- and FraCaS-derived rows and then draws 1,000 pairs proportionally over (phenomenon tag, label) cells with the same fixed seed. The L training pool draws on SICK TRAIN+TRIAL, the HANS pool minus the test sample, and HELP, capped at 6,000 pairs per source; a 600-pair stratified validation pool is split off before the cap is applied. The HANS non-entailment list draws, with a fixed seed, from the non-entailment subcases of the HANS evaluation file: 70 pairs from each of the three subcases that mirror the constructions of §6 (complement under a verb, passive, subject–object swap) and 30 from each of the other twelve, 570 pairs; it was built after the training pools and is not disjoint from them (below).

Non-entailment list. Table 6 gives the 570-pair HANS non-entailment list of §2 for every 235B condition with three seeds. All gold labels are unknown, so accuracy is one minus the share of

decisive verdicts, and every decisive verdict is a proof of an entailment that does not hold. The four constructions are those in which the premise contains a clause it does not assert: non-factive complements and non-factive adverbs (a lexical property), and garden-path complements and reduced relatives (a parsing error), 30–70 pairs each; the last column is the decisive share on the non-factive complements alone. The adapter row is judged after the adapter.

Table 6: HANS non-entailment list (570 pairs, gold unknown; 235B, mean±std over three seeds). Decisive: share of pairs given a yes or no verdict. Four constr.: share of decisive verdicts falling in the four constructions named in the text. Complement: decisive share on the 70 non-factive complement pairs. 127 of the 570 pairs occur in the label training or validation pools (Appendix A.6).

Condition	Accuracy	Decisive	Four constr.	Complement
L	$.758 \pm .009$	$.242 \pm .009$	$.944 \pm .018$	$.890 \pm .095$
L-bare	$.803 \pm .016$	$.197 \pm .016$	$.940 \pm .018$	$.743 \pm .052$
silver-view	$.801 \pm .023$	$.199 \pm .023$	$.926 \pm .027$	$.705 \pm .221$
S (silver)	$.802 \pm .018$	$.198 \pm .018$	$.894 \pm .005$	$.695 \pm .066$
written notation	$.844 \pm .038$	$.156 \pm .038$	$.906 \pm .032$	$.448 \pm .157$
refinement from the neutral prompt (one run)	.746	.254	.752	.557
L + adapter	$.752 \pm .004$	$.248 \pm .004$	$.939 \pm .018$	$.890 \pm .095$

Train–test integrity. Every arm is evaluated on the identical frozen lists; we verified that the evaluated pair-ID sets are equal across all conditions on the main model. No pair of the three test lists occurs in any training or validation pool (pair-level overlap is zero for every pool); the 570-pair non-entailment list shares 127 pairs with the label pools (122 training, 5 validation), on which the decisive shares of Table 6 do not differ from the rest. No prompt carries demonstrations, but the reflection LM does quote training sentences as worked examples inside the evolved instructions (0–16 per prompt across the evolved prompts of the grid and probes). Some of these sentences also occur in a test list, because SICK and HANS reuse sentences across pairs, but no pair does. Excluding the test pairs that contain a sentence quoted in any evolved prompt (120 SICK, 8 HANS, and 3 MED pairs, counting quoted spans of four or more words) changes no cell of Tables 2 and 3 by more than 0.8 points and no difference reported in §3 and §4 by more than 0.4 points.

A.7 Seed-paired tests behind §4

All tests are exact two-sided McNemar tests on paired per-pair correctness at 235B; "a/b" gives the discordant counts (first condition right and second wrong / the reverse).

Matched pool. Three-seed means: silver arm minus control $+.081$ / $+.024$ / $+.062$ on SICK / HANS / MED; silver-view minus control $+.089$ / $+.026$ / $+.063$. Silver-view over control: SICK $p < .001$ and MED $p \leq .04$ in every seed pairing; HANS mixed (one control win at $p = .046$). Silver-view against the silver arm over all nine seed pairings: SICK four silver-view wins, two silver-arm wins, three level (the seed spread of $\pm.03$ exceeds the $.008$ gap in means); HANS both ways; MED all $p > .3$. On seed 0 silver-view solves 85% / 94% / 86% of the pairs the silver arm solves, and on SICK 223 pairs are solved only by silver-view and 161 only by the silver arm. Surface-departure rate (content words of the sentence that appear neither as a predicate name nor as part of one in its formula; three seeds, pooled over sentences; script `analyze_surface_departure.py`): on SICK 4.3±1.4% / 9.4±1.8% / 9.4±1.8% for the silver, label, and bare arms; on MED 13.0±3.7% / 23.2±6.4% / 24.4±4.7%; on HANS every arm is between 1 and 3%. Parse failures on SICK: 0–8 pairs of 2,116 in L and L-bare cells against 160 and 232 under silver-view and silver reward (seed 0); mean nesting depth 5.2 against 2.2; parenthesis imbalance accounts for 84–87% of the silver-fed failures.

Written notation. $.511 \pm .025$ / $.919 \pm .006$ / $.644 \pm .048$. Against the control: SICK $p < 10^{-8}$ in every pairing; HANS two of three (the third at $p = .10$); MED two of three (one control win, $p = .044$). Against silver-view: SICK two silver-view wins and one notation win; HANS one notation win and two ties; MED silver-view above in two seeds ($p < 10^{-7}$ and $p = .057$) and level in one. Against the silver arm: SICK one win each and one tie; HANS one notation win and two ties; MED two silver-arm wins and one tie. Its second seed's MED entailment half is $.283$ against $.457$

and .362 for its other two seeds. Event notation in 85–100% of outputs. HANS constructions across seeds: understood objects .99–1.00 and passives .97–1.00 (L: .03–.14 and .90–.99); complement clauses under a verb .14–.33 (L: .91–1.00). Mean depth 3.0–3.4 (silver-view 3.0–5.4); parse failures 20–36 / 2–6 / 31–51 on SICK / HANS / MED (silver-view 47–160 / 13–28 / 52–78). False decisive verdicts on SICK containing an event with two subject equations: 26 of 34, 9 of 19, 46 of 53.

Adapter. On the control's outputs: HANS $+8.2$ / $+6.2$ / $+6.4$ points per seed (recovered / regressed 86/0, 66/1, 67/0), SICK $+5.2$ / $+2.9$ / $+2.8$ (125/15, 72/10, 66/6; McNemar $p < 10^{-11}$), MED $+1.1$ / $+0.2$ / -0.7 (20/9, 7/5, 1/8). On the label arm of §3: recovers 59–66 HANS pairs per seed with no regression, SICK $+3.3$ to $+3.7$ points, MED unchanged; on the non-entailment list the decisive count rises by 7, 3, and 1 pairs. Recovery shares on SICK, $(x - \text{control})/(\text{S} - \text{control})$: adapter 45%, silver-view 110%, written notation 120%; on MED 3%, 103%, 50%.

B Reference systems judged on identical pairs

We ran two reference systems through our judge on the same pairs as the 235B label arm: a hand-built symbolic pipeline (a three-parser CCG ensemble with WordNet abduction, in the line of Mineshima et al., 2015; Martínez-Gómez et al., 2016) and a ByT5-large (1.2B) parser we fine-tuned on the same 6,420 silver-annotated sentences as our silver arm (a pool drawn from SICK, HANS, and HELP). We do not compare them to our arms as methods: the model classes differ by two orders of magnitude, and each pipeline is a package with its own resource requirement (expert grammar, silver formulas, or labels). The lines below are reference points on identical pairs, scored strictly by the same rule (unknown never credited; the HANS and MED lists here are entailment-only, so strict equals recall).

System	SICK (2,116)	HANS (2,100)	MED (2,000)
symbolic pipeline	.612	.741	.474
ByT5-large, fine-tuned (1.2B)	.576	.808	.354
GEPA L (label), 235B, seed 0	.372	**.915**	.310
oracle of the three (symbolic, ByT5, L-235B)	.668	.990	.617

The SICK list is identical to Table 2's, so that cell is the seed-0 L run of the main grid; the HANS and MED lists are larger entailment lists (15 entailment subcases $\times$ 140; 2,000 MED entailments) and were translated with the same seed-0 L prompt. The MED list shares no pair with the label arm's training or validation pools. The HANS list shares 406 of its 2,100 pairs with them (HANS is template-generated); on the remaining 1,694 pairs the label arm scores .916, and both McNemar results hold on that subset. The lists are entailment-only by design. On a non-entailment pair the unknown verdict is the correct one, so a system is credited whenever its formulas prove nothing, including when they are wrong or fail to parse; on an entailment-only list every credited pair carries a proof, so the score measures translation the prover can use. On these two lists an error is an unknown verdict or, rarely, a proof of the negation (the false decisive counts below); false proofs on the non-entailment side are measured for every condition on the 570-pair HANS non-entailment list of §2 (Table 6). Exact McNemar tests on the paired records: on SICK the label arm is below both prior systems ($p < 10^{-7}$); on HANS it is above both (vs. ByT5 379 vs. 153, $p = 4 \times 10^{-23}$; vs. the symbolic pipeline 518 vs. 153, $p = 3 \times 10^{-47}$); on MED it is below both (vs. ByT5 $p = 1 \times 10^{-3}$). The ordering follows §3: the systems that hold a fixed compositional convention plus WordNet (symbolic) or weights fine-tuned on silver formulas win on lexical and monotonicity data, and the prompt-optimized translator wins where structural conventions decide. The errors are complementary rather than nested: the 235B label arm alone solves 71 SICK, 373 HANS, and 212 MED pairs that both prior systems miss, and a three-system oracle exceeds the best single system by 6, 7.5, and 14 points. False decisives remain rare on every system (SICK 2 / 0 / 10 and MED 10 / 2 / 3 for ByT5 / symbolic / L-235B).

C FOUR-MODEL GRID

Table 7 gives the seed-0 run of every condition for all four open-weight models and, for the two Qwen models, the seed-1 and seed-2 runs behind the three-seed means of Table 2. Ill-formed counts are pairs with at least one formula the parser rejects; false-decisive counts are decisive verdicts that contradict the gold label.

Table 7: Four-model grid: seed 0 of every condition, and seeds 1 and 2 of the three-seed conditions for the two Qwen models. Strict accuracy; ill-formed and false-decisive pair counts per set (SICK / HANS / MED).

Model	Condition	SICK	HANS	MED	ill-formed	false decisive
Qwen-235B	neutral prompt	0.166	0.290	0.522	21 / 38 / 81	2 / 0 / 7
	notation prompt	0.322	0.615	0.554	442 / 193 / 108	12 / 12 / 20
	L-bare	0.225	0.892	0.532	7 / 3 / 47	3 / 0 / 44
	L-bare, seed 1	0.287	0.875	0.605	0 / 1 / 10	2 / 0 / 23
	L-bare, seed 2	0.229	0.910	0.534	5 / 0 / 16	1 / 0 / 30
	L	0.372	0.898	0.594	8 / 2 / 36	10 / 0 / 67
	L, seed 1	0.358	0.901	0.571	6 / 2 / 28	8 / 0 / 32
	L, seed 2	0.373	0.928	0.615	4 / 0 / 25	7 / 0 / 37
	S (silver)	0.491	0.909	0.678	232 / 36 / 94	8 / 0 / 29
	S (silver), seed 1	0.518	0.910	0.674	160 / 33 / 82	4 / 0 / 43
	S (silver), seed 2	0.476	0.884	0.672	238 / 41 / 85	7 / 0 / 38
Qwen-32B	neutral prompt	0.090	0.098	0.515	176 / 56 / 85	8 / 0 / 2
	notation prompt	0.292	0.633	0.538	353 / 113 / 141	26 / 17 / 9
	L-bare	0.266	0.911	0.523	133 / 2 / 85	0 / 0 / 7
	L-bare, seed 1	0.191	0.841	0.512	40 / 1 / 42	0 / 0 / 4
	L-bare, seed 2	0.213	0.841	0.517	21 / 5 / 42	2 / 0 / 3
	L	0.362	0.832	0.553	47 / 4 / 81	9 / 0 / 13
	L, seed 1	0.337	0.826	0.592	4 / 1 / 23	12 / 0 / 34
	L, seed 2	0.310	0.885	0.561	61 / 0 / 32	9 / 0 / 11
	S (silver)	0.430	0.802	0.640	392 / 95 / 150	16 / 0 / 17
	S (silver), seed 1	0.445	0.748	0.593	278 / 150 / 347	16 / 0 / 21
	S (silver), seed 2	0.439	0.701	0.618	457 / 202 / 354	21 / 0 / 14
Llama-70B	neutral prompt	0.113	0.346	0.518	28 / 42 / 50	10 / 0 / 6
	notation prompt	0.152	0.465	0.514	846 / 142 / 334	6 / 0 / 7
	L-bare	0.187	0.770	0.539	25 / 6 / 51	4 / 0 / 17
	L	0.254	0.781	0.541	12 / 4 / 46	7 / 0 / 11
	S (silver)	0.371	0.742	0.597	784 / 154 / 264	6 / 0 / 23
Qwen-Coder-30B	neutral prompt	0.104	0.093	0.521	508 / 305 / 266	1 / 0 / 6
	notation prompt	0.229	0.381	0.529	572 / 240 / 305	14 / 0 / 12
	L-bare	0.182	0.617	0.516	338 / 192 / 246	4 / 0 / 18
	L	0.266	0.641	0.530	240 / 136 / 138	1 / 0 / 39
	S (silver)	0.355	0.742	0.563	462 / 184 / 193	12 / 1 / 49

D PROMPT-READING TABLES

Tables 8 and 9 support the construction-by-construction and axis-by-axis reading of §6. Both tables were coded by reading each evolved prompt against a fixed list of axes; a convention is marked only when the prompt states it or all of the run's outputs apply it.

Table 8: HANS constructions that separate the cells. Each construction has an accuracy row (its 70 test pairs, all gold entailments, so accuracy is recall) and a rule row (the rule the cell's prompt states for it), and two of them a false-yes row (235B seed-0 runs unless named; "adapter" is L's output re-judged after the reification adapter). *surface*: arity read off the sentence; *lexical*: arity read off the verb; *event*: roles are optional conjuncts; *agent-first*: argument order fixed by thematic role; *top-level*: complement asserted as a top-level conjunct; *scoped*: complement placed under a proposition variable; *none*: no rule for the construction (the frontier L prompt states no passive rule and keeps the agent-first order). *false yes*: rate of decisive verdicts on the construction's non-entailment mirror (70 pairs each from the HANS evaluation set, gold unknown: role-swapped passives; non-factive complements such as *believed that*), seed-0 rates; three-seed means are in Table 6. Bold marks the cells discussed in §6.

Construction		L	bare	S	silver-view	notation	terra L	llama L	adapter
Understood object	acc.	.14	**.69**	**1.00**	**1.00**	**.99**	.01	.00	**1.00**
	rule	surface	lexical	event	event	event	surface	surface	(code)
Passive	acc.	.90	.93	.97	1.00	.99	.97	**.00**	.90
	false yes	.04	.06	.01	.00	.00	—	—	—
	rule	agent-first	agent-first	event	event	event	none	surface	agent-first
Complement clause	acc.	.94	.81	.49	.30	.14	.57	.93	.94
	false yes	**.79**	**.69**	**.66**	**.70**	.29	—	—	—
	rule	none	none	none	none	none	scoped	top-level	none

Table 9: Representational axes across the label-only (L) seed triples, read from the prompts. A check marks a stated or consistently applied convention; text gives the choice where runs differ; *mixed*: the prompt states both policies.

Axis	235B s0 / s1 / s2			32B s0 / s1 / s2			Status
Entity = bound variable + noun predicate	✓	✓	✓	✓	✓	✓	fixed
Verb = flat n-ary predicate, subject first	✓	✓	✓	✓	✓	✓	fixed
Argument order by thematic role, not voice	✓	✓	✓	✓	✓	✓	fixed
Stated object kept (arity match)	✓	✓	✓	✓	✓	✓	fixed
Base-form lemma	✓	✓	✓	✓	✓	"pick one"	nearly fixed
Proper names as constants	no	yes	no	places only	no	mixed	open
Pronouns	`_person`	`_person`	literal	constant	—	—	open
Three-place predicates	allowed	—	forbidden	—	allowed	—	open
Prepositional phrases	relation	fused	fused	rel.+const	relation	relation	open
Understood object	1-place	1-place	1-place	—	—	1-place	surface where stated

E FAILURE TAXONOMY: FULL TABLES

Every wrong pair in the 60 evaluation cells of Appendix C (4 models × 5 conditions × 3 sets) is assigned to one of five classes by a deterministic script (`analyze_failure_taxonomy.py`) that reads the pair's formulas and the prover's diagnostics and tests the classes in the order listed, the first that applies being the class: *parse failure* (at least one formula rejected by the parser); *arity conflict* (the same predicate used with different argument counts in premise and hypothesis); *predicate-name variant* (the same content word rendered as different predicate names, including inflection and multiword differences); *lexical gap* (the hypothesis needs a lexical relation the premise and the WordNet axioms do not supply); *structural* (everything else: quantifier scope, negation, modeling choices). Tables 10–12 give the class shares per cell; Figure 3 plots the 235B rows and, for comparison, the silver-view seed-0 cell.

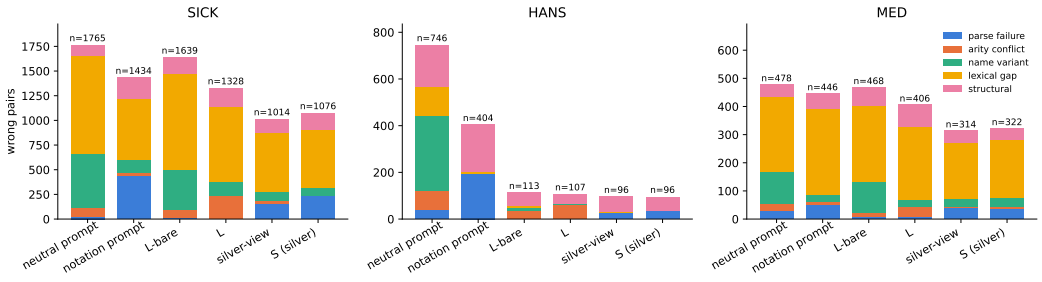


Figure 3: How the composition of errors changes with feedback (235B, seed 0). Each bar is one condition's wrong pairs on one set, split into the five failure classes of Appendix E; n is the number of wrong pairs. The silver-view condition is added for comparison and is not in the tables.

Table 10: Failure classes on SICK: wrong pairs per cell and the share of each class (%).

Model	Condition	wrong	parse fail.	arity	name var.	lexical gap	structural
Qwen-235B	neutral prompt	1765	1	5	31	56	6
	notation prompt	1434	31	2	9	42	15
	L-bare	1639	0	5	25	59	10
	L	1328	1	17	11	57	14
	S (silver)	1076	22	1	8	54	16
Qwen-32B	neutral prompt	1925	9	4	29	51	7
	notation prompt	1498	24	13	9	37	18
	L-bare	1554	9	9	19	52	11
	L	1351	3	15	19	52	11
	S (silver)	1206	33	3	9	46	10
Llama-70B	neutral prompt	1876	1	4	36	51	8
	notation prompt	1794	47	6	5	25	16
	L-bare	1720	1	7	27	52	13
	L	1579	1	5	36	55	3
	S (silver)	1332	59	2	6	29	5
Qwen-Coder-30B	neutral prompt	1896	27	4	19	41	9
	notation prompt	1631	35	4	6	36	19
	L-bare	1730	20	12	11	48	10
	L	1554	15	15	12	46	12
	S (silver)	1365	34	4	6	44	12

Table 11: Failure classes on HANS: wrong pairs per cell and the share of each class (%).

Model	Condition	wrong	parse fail.	arity	name var.	lexical gap	structural
Qwen-235B	neutral prompt	746	5	11	43	17	24
	notation prompt	404	48	0	0	2	50
	L-bare	113	3	28	12	7	50
	L	107	2	58	0	4	36
	S (silver)	96	38	0	0	1	61
Qwen-32B	neutral prompt	947	6	7	31	28	28
	notation prompt	385	29	0	1	3	67
	L-bare	93	2	74	1	3	19
	L	176	2	45	0	2	51
	S (silver)	208	46	0	0	0	54
Llama-70B	neutral prompt	687	6	9	42	8	35
	notation prompt	562	25	0	1	2	71
	L-bare	242	2	29	22	5	41
	L	230	2	30	45	0	23
	S (silver)	271	57	0	0	0	43
Qwen-Coder-30B	neutral prompt	952	32	5	14	19	30
	notation prompt	650	37	2	0	12	49
	L-bare	402	48	9	1	3	38
	L	377	36	8	3	3	50
	S (silver)	271	68	0	0	3	29

Table 12: Failure classes on MED: wrong pairs per cell and the share of each class (%).

Model	Condition	wrong	parse fail.	arity	name var.	lexical gap	structural
Qwen-235B	neutral prompt	478	6	5	24	55	9
	notation prompt	446	11	2	6	68	13
	L-bare	468	1	3	24	57	14
	L	406	2	9	6	63	19
	S (silver)	322	12	2	11	63	13
Qwen-32B	neutral prompt	485	8	8	30	46	7
	notation prompt	462	14	8	6	62	9
	L-bare	477	8	8	14	60	10
	L	447	6	9	12	61	12
	S (silver)	360	18	2	14	57	8
Llama-70B	neutral prompt	482	5	8	32	48	7
	notation prompt	486	31	3	7	51	8
	L-bare	461	3	4	27	59	6
	L	459	5	5	47	41	3
	S (silver)	403	34	3	15	42	5
Qwen-Coder-30B	neutral prompt	479	26	6	24	36	8
	notation prompt	471	31	8	4	48	9
	L-bare	484	26	6	14	45	9
	L	470	14	7	20	47	12
	S (silver)	437	26	6	7	48	13

F SINGLE-RUN PROBES

Table 13 gives the single-run cells discussed in §7, on the scales of Table 2.

Table 13: Single-run probe cells (strict). The MIPRO, TextGrad-style, and 235B-reflection rows use the 235B task model under the reward named in the row; the frontier rows use GPT-5.6-terra as the task model.

Condition	SICK	HANS	MED
L, MIPRO optimizer (1,940 pair evals)	.299	.780	.561
L, TextGrad-style optimizer	.380	.762	.581
S, TextGrad-style optimizer	.512	.862	.676
L, 235B as reflection LM	.283	.881	.560
S, 235B as reflection LM	.394	.719	.585
GPT-5.6-terra, L	.405	.888	.647
GPT-5.6-terra, S (silver)	.570	.932	.738

Zero-shot transfer. Table 14 gives every evolved prompt reused unchanged on FOLIO (203 pairs, multi-premise) and FraCaS (334 pairs; 183 with one premise, 151 with several), strict scoring, seed-0 prompts; the bare arm is judged without abduction. The all-unknown floor is .340 on FOLIO and .293 on FraCaS.

Table 14: Zero-shot transfer of the evolved prompts (strict), and the pair-level refinement of §5 applied to three of them. FraCaS is also split by the number of premises.

Model	Condition	FOLIO	FraCaS	FraCaS single	FraCaS multi
Qwen-235B	S (silver)	.345	.314	.366	.252
	L	.379	.422	.492	.338
	L-bare	.345	.383	.464	.285
	silver-view	.355	.428	.514	.325
	written notation	.365	.413	.486	.325
	L + refinement	.438	.613	.645	.573
	silver-view + refinement	.399	.548	.612	.470
	written notation + refinement	.414	.563	.590	.530
Qwen-32B	S (silver)	.345	.308	.355	.252
	L	.365	.404	.486	.305
	L-bare	.355	.362	.448	.258
Llama-70B	L	.360	.335	.388	.272
Qwen-Coder-30B	L	.350	.320	.361	.272

Refinement off-distribution. The three "+ refinement" rows of Table 14 run the refinement loop of §5 from the seed-0 formulas of L, silver-view, and the written notation, on the pairs the prover left unknown (at most three rounds). Silver-view and the written notation transfer at L's level on FraCaS (McNemar against L $p > .8$) and prove as many pairs as L does (84–102 decisive verdicts against the silver arm's 9). The refinement raises FraCaS from .422 to .613 and FOLIO from .379 to .438 for L, and the other two conditions by similar amounts ($p \leq .012$ for every cell). The gain is largest where the pair view has the most to add: FraCaS's multi-premise half rises by fifteen to twenty-four points, while FOLIO, whose pairs carry up to ten premises, gains four to six points and raises its parse failures two- to fivefold as the refinement rewrites every premise at once. On FraCaS's unknown pairs the false decisive count rises from 24 to 38 for L.

Re-optimization. Table 15 gives the two re-optimization probes of §7 (235B, one run per condition; budgets in Appendix A.1). On FOLIO the evolved-start deficit is 13 vs. 4 discordant pairs (McNemar $p = .049$); the neutral restart's edge over plain transfer is not significant ($p = .08$). On MED the two restarts have the same recall on entailment pairs (.443 vs. .453) and differ in false claims.

Table 15: Re-optimization on FOLIO (203 pairs) and MED (1,000 pairs), strict. "Decisive" is the number of yes/no verdicts; "false" the number of those that contradict the gold label.

	FOLIO			MED		
Condition	acc.	decisive	false	acc.	decisive	false
all-unknown floor	.340	0	0	.494	0	0
evolved L prompt, no re-optimization	.379	10	1	.594	233	67
re-optimized from the evolved L prompt	.374	14	5	.685	258	34
re-optimized from the neutral prompt	.419	23	4	.620	332	103