A Theorem-Proving-Based Evaluation of Neural Semantic Parsing

Hayate Funakura[1,2,3], Hyunsoo Kim[2], Koji Mineshima[2]

[1]Kyoto University [2]Keio University [3]Kikagaku Inc.

Correspondence: funakura.hayate.28p@st.kyoto-u.ac.jp

Abstract

Graph-matching metrics such as Smatch are the de facto standard for evaluating neural semantic parsers, yet they capture surface overlap rather than logical equivalence. We reassess evaluation by pairing graph-matching with automated theorem proving. We compare two approaches to building parsers: supervised fine-tuning (T5-Small/Base) and few-shot in-context learning (GPT-4o/4.1/5), under normalized and unnormalized targets. We evaluate outputs using graph-matching, bidirectional entailment between source and target formulas with a first-order logic theorem prover, and well-formedness. Across settings, we find that models performing well on graph-matching often fail to produce logically equivalent formulas. Normalization reduces incidental target variability, improves well-formedness, and strengthens logical adequacy. Error analysis shows performance degrades with increasing formula complexity and with coordination, prepositional phrases, and passive voice; the dominant failures involve variable binding and indexing, and predicate naming. These findings highlight limits of graph-based metrics for reasoning-oriented applications and motivate logic-sensitive evaluation and training objectives together with simplified, normalized target representations. All code and data for our experiments are publicly available.[1]

1 Introduction

Semantic parsing is the task of mapping natural language expressions into structured representations such as database queries or logical forms. These outputs have a wide range of applications, including document classification (Dong et al., 2015) and question answering (Yih et al., 2014). Neural network-based approaches have become prominent in semantic parsing (Konstas et al., 2017; Bai et al., 2022), with Smatch (Cai and Knight, 2013) widely used for evaluation. Smatch compares two Abstract Meaning Representations (AMRs) (Banarescu et al., 2013) by aligning their atomic propositions and computing the F-score of the overlap. We refer to Smatch and its variants as *graph-matching-based* evaluation.

The aim of this paper is to reconsider evaluation methods for neural semantic parsing from the perspective of logical reasoning. One key use of the logical forms produced by semantic parsing is to support logically correct inference. For such inference, the semantic parser must generate formal representations that enable a symbolic solver (e.g., an automated theorem prover) to derive correct outcomes such as entailment, contradiction, or consistency. Ideally, for a sentence S with a gold semantic representation $\mathrm{SR}_g(S)$, the parser's output $\mathrm{SR}_p(S)$ should be logically equivalent to $\mathrm{SR}_g(S)$. However, graph-matching-based evaluation focuses solely on surface overlap between graph components, and thus may fail to reflect whether the predicted and gold representations are truly equivalent in meaning.

In this paper, we test the hypothesis that a model achieving high performance in graph-matching-based evaluation does not necessarily perform well in evaluation aimed at accurate natural language inference. We evaluate models that convert English sentences into first-order predicate logic representations using both evaluation methods. We consider two model settings: supervised fine-tuning (SFT) of a Transformer-based pre-trained semantic parser, and in-context learning (ICL) with several pre-trained language models of different sizes, including the latest GPT-5, where a few parsing examples are given before parsing unseen sentences.

Our contributions are threefold:

[1] https://github.com/hfunakura/text2sem

Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP, pages 295–306

Limits of graph-based metrics Pairing graph-matching with theorem-prover entailment reveals a gap between surface overlap and logical equivalence: models with high graph-matching scores often fail to produce logically correct predictions.

Benefit of target normalization Training on normalized formulas consistently improves performance by reducing incidental variability and enhancing well-formedness and logical adequacy.

Error patterns and next steps Performance declines with formula complexity and with coordination, prepositional phrases, and passive voice. The most frequent errors involve variable binding and indexing and predicate naming, motivating stronger handling of linguistic phenomena and simpler target representations.

2 Background and Related Work

2.1 Semantic Parsing

The task of converting natural language expressions into formal semantic representations has been extensively studied in the fields of symbolic logic and formal semantics (Blackburn and Bos, 2005; Lepore and Cumming, 2009). Advances in this area have been accelerated by the development of syntactically expressive grammar formalisms such as Combinatory Categorial Grammar (CCG) (Steedman, 2000), the creation of linguistically rich resources like CCGbank (Hockenmaier and Steedman, 2007), and the emergence of wide-coverage semantic parsers enabled by progress in syntactic parsing technologies (Bos et al., 2004).

More recently, the development of semantically annotated corpora such as AMR (Banarescu et al., 2013) and Parallel Meaning Bank (PMB) (Abzianidze et al., 2017) has accelerated research into neural approaches to semantic parsing. In particular, sequence-to-sequence models have become widely adopted for learning mappings from natural language to logical forms. These models have been successfully applied to a variety of downstream tasks, including code generation (Ling et al., 2016), question answering (Dong and Lapata, 2016), and natural language generation (Konstas et al., 2017).

Originally, semantic parsing was considered a promising approach for enabling and improving a wide range of downstream tasks requiring semantic understanding, including translation, summarization, question answering, and paraphrasing. Graph-matching-based evaluation methods, such as Smatch (Cai and Knight, 2013) and subsequent variants for AMR (Opitz et al., 2020; Opitz, 2023), as well as adaptations for Discourse Representation Structure (DRS) (van Noord et al., 2018), were developed with this broad applicability in mind and provide flexible means to assess parsing performance across diverse use cases. However, with the rise of large pre-trained language models (Devlin et al., 2019; Brown et al., 2020), the role of formal semantic representations in these downstream tasks has lessened, and the relevance of semantic parsing itself has come under renewed scrutiny (van Noord et al., 2020).

2.2 Logical Entailment

One core task where formal semantic representations play a role is recognizing logical entailment in natural language inference, typically formulated as determining whether a premise entails, contradicts, or is neutral with respect to a hypothesis. Traditional logical formalisms developed within symbolic logic (Blackburn and Bos, 2005) were often designed with the goal of enabling precise logical reasoning—such as entailment and consistency checking—by combining these representations with Automated Theorem Proving (ATP) techniques (Fitting, 1996; Robinson and Voronkov, 2001). An early attempt to directly apply this paradigm to natural language entailment recognition was proposed by Bos and Markert (2005). Further work has extended this line of research by leveraging semantic parsing based on CCG in combination with theorem provers to handle a wide range of natural language inferences (Abzianidze, 2015; Mineshima et al., 2015; Haruta et al., 2022). In contrast, the effectiveness of combining neural semantic parsing with automated theorem proving for natural language inference remains, to the best of our knowledge, an open question that has not yet been fully explored.

One of the domains where precise logical reasoning is essential is mathematical theorem proving. The task of converting natural language proofs into formal representations that can be handled by automated theorem provers or interactive proof assistants such as Coq and Lean has been extensively studied under the name of *autoformalization* (Wu et al., 2022). Whether pretrained language models alone can support the kind of rigorous logical reasoning required for

complex problem solving remains an open question. At present, there is ongoing exploration into hybrid approaches that integrate structured, logic-based semantic representations, which guarantee precision and correctness, with statistical language models (Kautz, 2022).

2.3 Compositional Generalization

Another domain where neural semantic parsing has been actively studied is compositional generalization, which examines whether a model can generalize to novel syntactic and semantic combinations when mapping natural language expressions to logical forms. A widely used benchmark in this area is COGS (Kim and Linzen, 2020), followed by numerous extensions, most of which rely on exact matching of predicted and gold logical forms as the evaluation metric. However, exact matching treats logically equivalent expressions as different—for example, it fails to recognize the equivalence of conjunctive forms such as $p \wedge q$ and $q \wedge p$, or of formulas that differ only in variable naming, such as $\exists x_1(\mathtt{cat}(x_1) \wedge \mathtt{run}(x_1))$ and $\exists x_7(\mathtt{cat}(x_7) \wedge \mathtt{run}(x_7))$.

To address these shortcomings, successor benchmarks such as ReCOGS (Wu et al., 2023) and SLOG (Li et al., 2023) adopt graph-matching-based metrics similar to Smatch, which account for permutations of conjuncts and variable renaming. ReCOGS in particular generates multiple logically equivalent variants of each target logical form, demonstrating that even minor surface differences (e.g., variable names or parentheses) can substantially affect evaluation. To our knowledge, however, none of these studies have employed a theorem prover to verify logical equivalence. A theorem prover naturally subsumes permutations of conjuncts and variable renaming as part of logical equivalence, and moreover is capable of handling richer equivalences involving, for example, negation and nested quantifiers.

Building on this background, the present study examines the capabilities of current neural semantic parsing models through evaluation with automated theorem proving. This line of inquiry is intended to lay the groundwork for developing models that achieve greater precision in natural language reasoning.

3 Experimental Setup

This section describes the experimental setups for both SFT and ICL conducted in this study.

3.1 Dataset

We build our dataset from SICK (Marelli et al., 2014), a benchmark for natural language inference. SICK consists of about 10,000 sentence pairs, each consisting of a premise (p) and a hypothesis (h), annotated with graded semantic relatedness (ranging from 1 to 5) as well as an entailment label chosen from $\{\mathtt{entailment}, \mathtt{contradiction}, \mathtt{neutral}\}$. It includes linguistic phenomena such as quantification and negation, making it suitable for evaluating logically complex semantic representations.

We use its original train/test split for training and evaluation, strictly following the official division in all experiments. The training set contains 4,500 sentence pairs (9,000 sentences in total), and the test set contains 4,927 sentence pairs (9,854 sentences in total).

To obtain semantic representations that faithfully reflect sentence meaning, we parse all SICK sentences using ccg2lambda (Mineshima et al., 2015; Martínez-Gómez et al., 2016). In this system, a CCG parser is first applied to produce a CCG derivation tree, which is then mapped into a logical form (semantic representation) via standard λ-calculus–based semantic composition; in our experiments, we use depccg (Yoshikawa et al., 2017) as the CCG parser. Given a premise p and a hypothesis h, the system outputs their semantic representations $\mathrm{SR}(p)$ and $\mathrm{SR}(h)$, and then employs a theorem prover, together with axioms derived from external knowledge bases such as WordNet (Fellbaum, 1998) to determine whether $\mathrm{SR}(p)$ entails $\mathrm{SR}(h)$, contradicts it, or neither. The output is thus one of three labels: `entailment`, `contradiction`, or `neutral`.

For the target representation, we adopt event semantics (Parsons, 1990), a framework widely used in semantic parsing (e.g., in COGS (Kim and Linzen, 2020)), and employ the event-semantics templates for ccg2lambda developed by Martínez-Gómez et al. (2017). Table 1 illustrates example sentences from SICK together with their event-semantic representation, where (b) shows the raw representations produced by ccg2lambda.

To obtain high-quality sentence-formula pairs, we filter SICK sentence pairs as follows. We

Table 1: Example sentences and their event-semantic representations. In the IDs, "p" indicates the premise and "h" indicates the hypothesis. In the semantic representations, variables of the form $x_1, x_2, \ldots$ are used for entities, while $e_1, e_2, \ldots$ are used for events. The hyphen "-" denotes logical (boolean) negation. "Complexity" counts the number of logical constants (negation, quantifier, and conjunction). "`exists e1 x2 x3`" is an abbeviation for "`exists e1. exists e2. exists e3`".

ID	`sick_train_88_p`	Complexity
(a) Sentence	There is no biker jumping in the air.	—
(b) Raw	`-exists x5.(_biker(x5) & exists e6.(_jump(e6) & (subj(e6) = x5) & exists x7.(_air(x7) & _in(e6,x7))))`	8
(c) Prenex	`-exists e1 x2 x3.(biker(x2) & jump(e1) & (subj(e1) = x2) & air(x3) & in(e1,x3))`	8
ID	`sick_train_55_p`	
(a) Sentence	Three boys are jumping in the leaves.	—
(b) Raw	`exists x4.(_boy(x4) & _three(x4) & exists e5.(_jump(e5) & (subj(e5) = x4) & exists x6.(_leaf(x6) & _in(e5,x6))))`	8
(c) Prenex	`exists e1 x2 x3.(boy(x2) & three(x2) & jump(e1) & (subj(e1) = x2) & leaf(x3) & in(e1,x3))`	8

retain only those with a gold SICK label in $\{\texttt{entailment}, \texttt{contradiction}\}$ for which the theorem prover's judgment over the ccg2lambda-derived representations matches the gold label. Pairs labeled `neutral` are excluded, since such outcomes may occasionally arise from pipeline errors (e.g., parsing failures or misaligned logical forms), making it difficult to guarantee correct semantic representations. From the retained pairs, we pair each sentence with its event-semantics formula to construct the training and evaluation instances. The resulting dataset comprises 2,392 training examples and 2,580 test examples.

We use two types of target representations to enable comparison with simplified formulas. Table 1 presents concrete examples of both representations, shown in (b) and (c), respectively.

1. Raw ccg2lambda outputs: The unmodified output of ccg2lambda, represented as first-order predicate logic with event variables. In these formulas, quantifiers (particularly existential quantifiers) may appear in different positions depending on the sentence structure, and variable names are assigned arbitrarily.

2. Prenex-normalized formulas (PNF): Derived deterministically from the raw `ccg2lambda` outputs by (i) moving all quantifiers to the sentence prefix with systematic variable renaming (while keeping negations at the front of the sentence), (ii) normalizing predicate symbols (e.g., removing leading underscores introduced by `ccg2lambda`), and (iii) reassigning variable indices starting from 1. This normalization reduces incidental variation in quantifier placement and superficial symbol noise that could otherwise confound sequence models.

We include both variants to assess the effect of target-side standardization via prenex normalization on model performance.

3.2 Annotation

We introduced additional categories to assess which natural language phenomena present challenges for neural semantic parsing. Specifically, using the output of the CCG parser, we annotated the SICK dataset with three syntactic categories:

- **Coordinating Conjunctions (CC):** Sentences containing coordinating conjunctions such as "and" and "or", which require proper handling of coordination scope.
- **Prepositional Phrases (PP):** Sentences with prepositional phrases that introduce spatial, temporal, or relational information requiring compositional analysis.
- **Passive Voice (PSS):** Sentences exhibiting passive voice constructions where argument structure differs from canonical active voice.

The annotation was conducted using automated extraction from CCG parse trees. We developed search scripts to identify these phenomena from syntactic categories and logical formulas patterns.

Table 2: Example sentences annotated by category. #Train indicates the number of occurrences in the SICK train data and #Test indicates the number of occurrences in the SICK test data.

Category	#Train	#Test	Example
Conj	1846	2065	There is no dog wrestling and hugging. (`sick_train_13_h`)
PP	1500	1679	A little girl is looking at a woman in costume. (`sick_train_74_p`)
Passive	795	839	Children covered by leaves are playing with red shirts. (`sick_train_61_p`)

Table 3: DRSs converted from the examples in Table 1: `sick_train_88_p` (left) and `sick_train_55_p` (right).

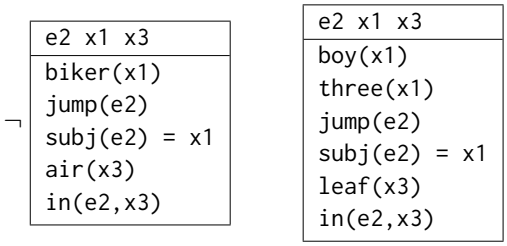


Table 2 shows the distribution of these categories in our dataset, with CC being the most frequent (2,790 instances), followed by PP (2,458 instances) and PSS (1,519 instances). This annotation enables us to analyze parser performance across different linguistic phenomena and identify which categories are particularly challenging for neural approaches.

3.3 Supervised Fine-tuning

To construct the semantic parser via SFT, we adopted T5-Small and T5-Base (Raffel et al., 2020) as pretrained models. Training was conducted for 50 epochs with a batch size of 16, a learning rate of 1×10^{-5}, weight decay of 0.01, and a warm-up of 500 steps. The maximum input and output sequence lengths were set to 256 tokens each. For each model–task combination, we trained and evaluated with seeds 1–10.

3.4 In-context Learning

We conducted few-shot in-context semantic parsing with large language models. For this experiment, we selected GPT-4o, GPT-4.1, and GPT-5, representing state-of-the-art models of different sizes that are widely used for reasoning-oriented tasks. The prompt comprised five text–formula exemplars randomly sampled from the training set, together with basic conventions for the target formalism (e.g., notation for existential quantification and negation, and predicate-naming conventions for compound expressions). The full prompt is provided in Appendix A.

For all models, the temperature was set to 0.0. To account for variability, we ran each configuration with random seeds fixed at 1, 2, and 3. For GPT-5, we additionally set the API parameters `reasoning.effort` (a budget indicator for the reasoning phase) and `text.verbosity` (the verbosity of the textual response) to `minimal`.

For cost considerations related to API usage, we restricted ICL to prenex-normalized data and did not include raw ccg2lambda outputs, and we evaluated each model in a single run.

3.5 Evaluation Metrics

We compare the semantic parsing outputs of each method using two evaluation metrics: graph-matching and automated theorem proving.

For the graph-matching evaluation, we use Counter (van Noord et al., 2018).[2] Counter is a modification of Smatch for graph structures in which scope-taking phenomena such as negation and quantification matter. Because Counter supports DRSs, we convert the parser's predictions (i.e., FOL formulas in event semantics) into DRSs and use Counter to compute the F-score between the gold DRS and the predicted DRS. We refer to this F-score, together with its components precision and recall, as *Dmatch*. The conversion from FOL to DRS was performed using the conversion script provided in ccg2lambda.[3] For example, the two formulas in Table 1 are converted into DRSs, as shown in Table 3. [4]

In addition, we evaluate via automated theorem proving. We use Vampire (Kovács and Voronkov, 2013), a state-of-the-art first-order theorem prover.[5] To examine whether the parser's

[2] https://github.com/RikVN/DRS_parsing

[3] https://github.com/mynlp/ccg2lambda

[4] The indices of the variables for entities and events are assigned according to the order of their occurrences in the Raw formula in Table 1.

[5] https://github.com/vprover/vampire

Table 4: SFT results (raw ccg2lambda outputs): mean ± standard deviation (n=10)

Model	Exact Match	Prover Acc	Dmatch Precision	Dmatch Recall	Dmatch F1	Non-WFF Ratio
T5-Small	0.101 ± 0.003	0.189 ± 0.005	0.611 ± 0.011	0.504 ± 0.010	0.544 ± 0.010	0.240 ± 0.012
T5-Base	0.322 ± 0.002	0.634 ± 0.004	0.887 ± 0.004	0.864 ± 0.004	0.873 ± 0.004	0.031 ± 0.003

Table 5: SFT results (prenex-normalized formulas): mean ± standard deviation (n=10)

Model	Exact Match	Prover Acc	Dmatch Precision	Dmatch Recall	Dmatch F1	Non-WFF Ratio
T5-Small	0.411 ± 0.003	0.439 ± 0.002	0.771 ± 0.002	0.739 ± 0.002	0.752 ± 0.002	0.018 ± 0.002
T5-Base	0.674 ± 0.004	0.689 ± 0.004	0.889 ± 0.002	0.874 ± 0.002	0.880 ± 0.002	0.007 ± 0.001

prediction is logically equivalent to the gold reference, we test whether bidirectional entailment holds between the gold formula and the prediction.

Dmatch provides a similarity score based on clause overlap in DRSs, but it does not reveal the precise logical relation between two formulas (e.g., equivalence, entailment, or contradiction). For example, consider the following pairs:

$g_1 =$ `exists e.jump(e)`
$p_1 =$ `exists e.(jump(e) & high(e))`

$g_2 =$ `exists e x.(eat(e) & (subj(e)=x))`
$p_2 =$ `exists e x.(eat(e) & (obj(e)=x))`

Both $\langle g_1, p_1 \rangle$ and $\langle g_2, p_2 \rangle$ receive the same Dmatch score of 0.5, even though the first reflects one-way entailment (the prediction p_1 overspecifies the gold g_1) while the second is a clear semantic role mismatch with no entailment. Similarly, the pair of logically unrelated formulas `P(a)` and `Q(a)`, and the contradictory pair `P(a)` and `-P(a)`, both receive a Dmatch score of 0 and are thus indistinguishable. Since such distinctions are essential for reasoning tasks, we complement graph-based evaluation with theorem proving, which explicitly identifies logical relations.

4 Results

We report results for SFT and few-shot ICL on our semantic parsing task, evaluated using three criteria: graph-matching (Dmatch), theorem-prover–based equivalence, and well-formedness. All metrics are reported to three decimal places (rounded half-up).

4.1 Supervised Fine-tuning

For SFT, we trained with random seeds 1–10 on two target representations (raw formulas and prenex formulas) and evaluated on the test split. Tables 4 and 5 report the results.

Across both representations, T5-Base shows higher values than T5-Small. The gaps in Prover Accuracy and Dmatch F1 are present in both settings and are larger on raw ccg2lambda outputs; the Non-WFF Ratio is lower for T5-Base in both settings, especially on raw.

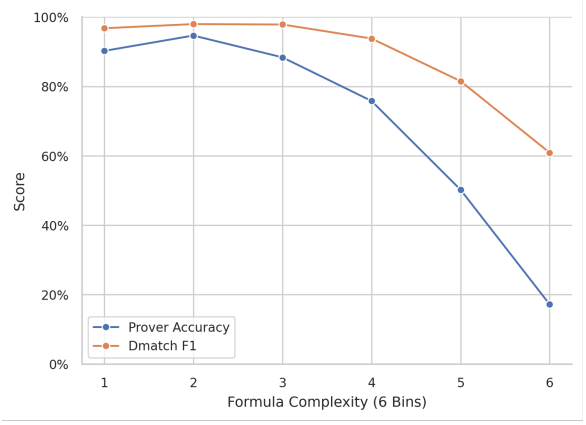
Within each model, prenex normalization is associated with higher Prover Accuracy and Exact Match and a lower Non-WFF Ratio. This pattern indicates that suppressing incidental variability in the target-side quantificational structure is associated with better parser performance.

The results therefore point to concrete directions for improving semantic parsers: first, leverage natural scaling with model size—performance increases consistently from T5-Small to T5-Base across both representations and metrics—indicating that greater capacity is a straightforward path to better end-to-end behavior; second, normalize the target representation via PNF to suppress incidental variability, reduce the Non-WFF ratio, and improve alignment with theorem-proving-based evaluation.

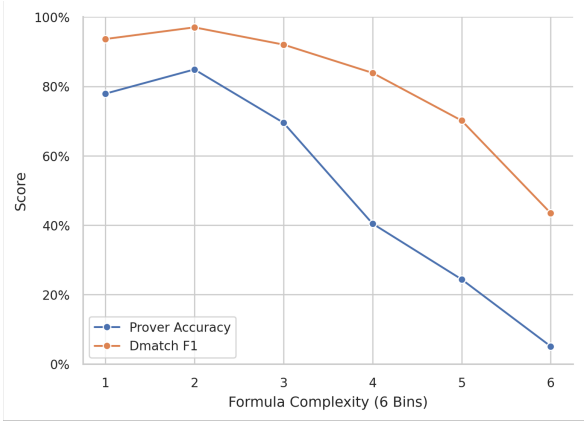
Looking across both representations and both models, higher Dmatch F1 does not translate into commensurately high Prover Accuracy. For raw ccg2lambda outputs, T5-Small attains Dmatch F1 of 0.544, whereas Prover Accuracy is 0.189; even T5-Base shows 0.873 and 0.634, respectively. With prenex normalization the gap narrows but remains (0.752 and 0.439 for T5-Small; 0.880 and 0.689 for T5-Base). This consistent disparity indicates that clause-level graph-matching captures structural overlap rather than the logical equivalence, underscoring the need for evaluation that are sensitive to logical structure.

4.2 In-context Learning

All ICL results are obtained on prenex-normalized targets and averaged over three runs with random



(a) T5-Base, seed=5



(b) GPT-5, seed=1

Figure 1: Relationship between target-side formula complexity and model performance. (a) best SFT configuration (T5-Base, seed 5); (b) best ICL configuration (GPT-5, seed 1).

Table 6: ICL results (prenex-normalized formulas): mean ± standard deviation (n=3)

Model	Exact Match	Prover Acc	Dmatch Precision	Dmatch Recall	Dmatch F1	Non-WFF Ratio
GPT-4o	0.328 ± 0.029	0.493 ± 0.046	0.824 ± 0.017	0.812 ± 0.026	0.816 ± 0.022	0.012 ± 0.006
GPT-4.1	0.278 ± 0.046	0.474 ± 0.034	0.734 ± 0.027	0.737 ± 0.033	0.733 ± 0.030	0.083 ± 0.025
GPT-5	0.318 ± 0.037	0.514 ± 0.009	0.806 ± 0.013	0.803 ± 0.010	0.803 ± 0.011	0.010 ± 0.002

seeds 1, 2, and 3 (Table 6). GPT-5 achieves the best overall scores (Exact Match 0.318; Prover Accuracy 0.514; Dmatch F1 0.803). GPT-4o follows (0.328; 0.493; 0.816), and GPT-4.1 shows lower Dmatch F1 (0.733) and the highest Non-WFF Ratio (0.083). Both GPT-4o and GPT-5 maintain a very low Non-WFF Ratio (0.012 and 0.010), indicating that the prompt conventions yield mostly well-formed formulas.

Relative to SFT on prenex targets (Table 5), even the strongest ICL setting (GPT-5) trails T5-Base in Prover Accuracy (0.514 compared with 0.689) and Exact Match (0.318 compared with 0.674), while matching well-formedness (Non-WFF Ratio 0.010 compared with 0.007). Dmatch F1 is lower than T5-Base (0.803 compared with 0.880) but remains in a similar range; as noted in the SFT results, graph-matching scores tend to run higher than prover-based equivalence.

5 Discussion and Future Perspectives

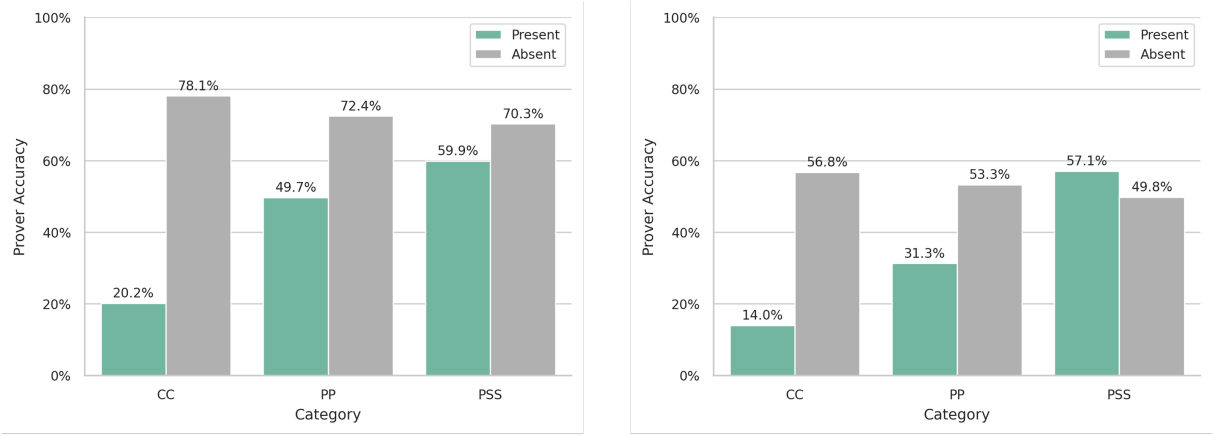
In this section, we focus our analysis on the strongest configuration within SFT—T5-Base (seed 5, final checkpoint)—and, for ICL, we use GPT-5 with seed 1 as a representative setting. All results are restricted to prenex-normalized targets. We begin by examining how target-side formula complexity modulates performance, then analyze the impact of syntactic and semantic phenomena (coordinating conjunctions, prepositional phrases, passive voice), and finally investigate the sources of mispredictions through a category-wise breakdown centered on the overall best model, T5-Base.

5.1 Impact of Formula Complexity

We grouped test instances by the complexity of the target-side formula and computed Prover Accuracy and Dmatch for each bin. Formula complexity was measured by counting logical constants, specifically negation, quantifiers, and conjunctions (see Table 1 for examples). Instances were sorted by complexity and split into six equal-sized groups. In our test set ($n = 2,580$), each group therefore contains 430 instances. Results in Figure 1 show a clear performance drop as complexity increases, with T5-Base consistently outperforming GPT-5 across all bins. Notably, even GPT-5, despite its ability to handle very long outputs (up to 128k tokens), remains sensitive to structural complexity, suggesting that future work should focus on improving robustness to long and compositional formulas.

5.2 Impact of Syntactic Features

We conducted a stratified analysis by linguistic phenomena, comparing Prover Accuracy between the presence and absence of coordination, preposi-



(a) T5-Base, seed=5

(b) GPT-5, seed=1

Figure 2: Prover accuracy stratified by the presence or absence of syntactic features. X-axis labels follow Section 3.2. (a) best SFT configuration (T5-Base, seed 5); (b) best ICL configuration (GPT-5, seed 1).

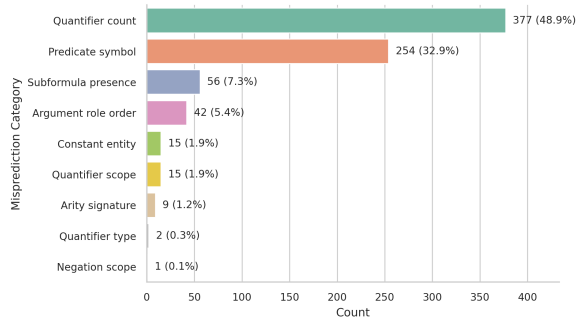


Figure 3: Error type distribution of 771 prover-failed predictions by T5-Base.

tional phrases, and passive voice. Figure 2 shows the results. Table 7 in Appendix B shows error examples of each category. The results show that coordination is a major source of difficulty in both settings: its presence is associated with a large reduction in Prover Accuracy relative to its absence. Prepositional phrases also correlate with decreased accuracy, reflecting the burden of resolving relational structure and attachment in a way that remains faithful to subsequent formal inference. By contrast, passive voice exhibits divergent behavior across models: one model appears less sensitive or even slightly advantaged by passive constructions, whereas the other shows decreased accuracy under passive voice. A plausible interpretation is that the prompt and large pretraining may induce templates that better regularize argument-structure alternations, while fine-tuned parameters benefit more from canonical (active) realizations emphasized during training.

5.3 Breakdown of Mispredictions

Among the predictions of T5-Base, the best overall model, we selected 771 cases that were well-formed but failed the prover-based evaluation. For fine-grained error analysis, these cases were presented individually to an LLM (GPT-4o), which was instructed to assign a label corresponding to the most critical problem in each prediction. The set of labels was predefined by the authors, and the LLM selected one from this set. In total, we prepared 11 label types, including, for example, Predicate Symbols (predicate mismatches such as `loud` vs. `loudly`), Subformula Presence (missing subformulas), and Argument Role Order (incorrect variable order in `subj` and `obj` functions).

The results are summarized in Figure 3. The two most frequent error types, quantifier-count mismatch and predicate-name error, together account for 81.8% of all mispredictions. The most frequent category, quantifier-count mismatch, accounts for 48.9% overall. Table 8 in Appendix C shows two representative examples. These errors are not superficial notation issues; they reflect fundamental misinterpretations of the semantic structure of the source text. Addressing them requires strategies that tie natural language understanding more tightly to the intended semantic framework (here, event semantics) during prediction.

5.4 Outlook Based on the Behaviors of SFT and ICL

Based on the respective behaviors of SFT and ICL, a coherent picture emerges: parameter learning aligns outputs more tightly with theorem-

prover criteria, whereas prompt-only conditioning of foundation models yields consistently well-formed formulas under prenex conventions yet comparatively weaker semantic alignment. This suggests a practical division of labor—use ICL as a high–well-formedness candidate generator, paired with a component optimized for semantic adequacy (e.g., a fine-tuned parser or a prover-guided reranker/repair module). Such a hybrid pipeline can narrow the remaining gap without sacrificing the strengths of either approach.

6 Conclusion

We propose evaluation with an automated theorem prover as a precise test of whether a neural semantic parser faithfully captures a sentence's semantic structure. Coupled with graph-matching, it reveals a persistent gap between surface overlap and logical equivalence across supervised fine-tuning and in-context learning, with normalization improving well formedness and logical adequacy. Sentences with coordination, prepositional phrases, or passive voice are more error prone, and errors concentrate in variable binding and indexing and in predicate naming. Looking ahead, a hybrid strategy that pairs generation with logic-aware verification is a promising direction.

7 Limitations

Although we employ popular approaches such as SFT with T5 and ICL with OpenAI models, we have not, for example, fine-tuned billion-parameter variants or evaluated architectures with substantially different design principles; accordingly, the applicability of our conclusions is limited. Additionally, we use OpenAI models via API access, which limits transparency into the models and may constrain long-term reproducibility.

Our analysis is conducted on SICK, a natural language inference benchmark that is widely used for studying entailment relations involving compositional logical structures such as negation and quantification; we chose this dataset for precisely these properties, while noting that examining additional datasets and domains will help assess generality.

Another limitation of this study is that we do not evaluate end-to-end natural language inference, one of the principal downstream tasks. Instead, we focus on bidirectional entailment between predicted and gold logical forms, since this directly serves our objective of assessing the formulas themselves.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments and suggestions. This work is partially supported by JST, CREST Grant Number JPMJCR2114.

References

Lasha Abzianidze. 2015. A tableau prover for natural logic and language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2492–2502, Lisbon, Portugal.

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247.

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. *arXiv preprint arXiv:2203.07836*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Patrick Blackburn and Johan Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI.

Johan Bos, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th international conference on Computational Linguistics*, pages 1240–1246.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 628–635.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language

models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43.

Li Dong, Furu Wei, Shujie Liu, Ming Zhou, and Ke Xu. 2015. A statistical parsing framework for sentiment classification. *Computational Linguistics*, 41(2):293–336.

Christiane Fellbaum. 1998. *WordNet: An electronic lexical database*. MIT press.

Melvin Fitting. 1996. *First-Order Logic and Automated Theorem Proving*. Springer.

Izumi Haruta, Koji Mineshima, and Daisuke Bekki. 2022. Implementing natural language inference for comparatives. *Journal of Language Modelling*, 10(1):139–191.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Henry A Kautz. 2022. The third AI summer: AAAI Robert S. Engelmore Memorial Lecture. *AI Magazine*, 43(1).

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157.

Laura Kovács and Andrei Voronkov. 2013. First-order theorem proving and vampire. In *International Conference on Computer Aided Verification*, pages 1–35. Springer.

Ernest Lepore and Sam Cumming. 2009. *Meaning and Argument: An Introduction to Logic Through Language*. Wiley-Blackwell.

Bingzhi Li, Lucia Donatelli, Alexander Koller, Tal Linzen, Yuekun Yao, and Najoung Kim. 2023. SLOG: A structural generalization benchmark for semantic parsing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3213–3232, Singapore. Association for Computational Linguistics.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: A compositional semantics system. In *Proceedings of ACL-2016 System Demonstrations*, pages 85–90.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2017. On-demand injection of lexical knowledge for recognising textual entailment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 710–720, Valencia, Spain. Association for Computational Linguistics.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2061.

Juri Opitz. 2023. SMATCH++: Standardized and extended evaluation of semantic graphs. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1595–1607, Dubrovnik, Croatia. Association for Computational Linguistics.

Juri Opitz, Letitia Parcalabescu, and Anette Frank. 2020. AMR similarity metrics from principles. *Transactions of the Association for Computational Linguistics*, 8:522–538.

Terence Parsons. 1990. *Events in the Semantics of English*. MIT Press, Cambridge, MA.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Alan Robinson and Andrei Voronkov. 2001. *Handbook of Automated Reasoning*, volume 1. Elsevier.

Mark J. Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge.

Rik van Noord, Lasha Abzianidze, Hessel Haagsma, and Johan Bos. 2018. Evaluating scoped meaning representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA).

Rik van Noord, Antonio Toral, and Johan Bos. 2020. Character-level representations improve DRS-based semantic parsing even in the age of BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4587–4603.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368.

Zhengxuan Wu, Christopher D. Manning, and Christopher Potts. 2023. ReCOGS: How incidental details of a logical form overshadow an evaluation of semantic interpretation. *Transactions of the Association for Computational Linguistics*, 11:1719–1733.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 643–648.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 277–287, Vancouver, Canada. Association for Computational Linguistics.

A Prompt Template for In-context Learning

Below is the exact prompt template used for semantic parsing with ICL (described in Section 3.4). Placeholders are enclosed in braces.

```
System message:
You are a precise semantic parser that maps
    natural language to a logical formula.
    Respond with only the formula, no
    explanations.

User message:
Examples:

text: <exemplar_1_text>
formula: <exemplar_1_formula>

text: <exemplar_2_text>
formula: <exemplar_2_formula>

text: <exemplar_3_text>
formula: <exemplar_3_formula>

text: <exemplar_4_text>
formula: <exemplar_4_formula>

text: <exemplar_5_text>
formula: <exemplar_5_formula>

Guidelines for this task:
Output only the logical formula, no explanations
    .
Use plain predicate and role names (no leading
    underscores) exactly as in the examples:
dog(x1), run(e1), in(e1,x3), (subj(e1) = x1), (
    obj(e1) = x2).
Quantification: exists e x.(...). Conjunction:
    &. Equality: =.
Negation: use the hyphen '-'.
Multiword predicates are single tokens joined
    with underscores: in_front_of(e,x).
Variables: entities x1,x2,...; events e1,e2,....
     Keep parentheses balanced and whitespace
    minimal.

Now parse the following text to its logical
    formula.

text: {<SOURCE_TEXT>}
formula:
```

B Error Examples by Syntactic Features

Table 7 shows error examples from the T5-Base model for each syntactic feature.

C Examples of Quantifier Count Errors

Table 8 presents examples of Quantifier Count errors.

In `sick_test_230_h`, two distinct event variables should be bound, but only one is introduced; the playing and waiting events are collapsed into a single event, and the variable `x3` is bound without appearing in the body.

In `sick_test_2872_p`, the predicate `mechanical` should take the same variable as `bull`, but it incorrectly takes a different variable, leading to a mismatch in argument linkage.

Table 7: Examples of typical errors in neural semantic parsing categorized by syntactic features.

Error Type	Example (Gold vs. Predicted)
Coordinating Conjunctions (CC)	**Sentence:** A man [$_{\text{CONJ}}$ and] a woman are sitting comfortably on the bench. (sick_test_706_p) **Gold:** exists e1 e2 x3 x4 x5 x6.(man(x3) & sit(e1) & (subj(e1) = x3) & comfortably(e1) & bench(x4) & on(e1,x4) & woman(x5) & sit(e2) & (subj(e2) = x5) & comfortably(e2) & bench(x6) & on(e2,x6)) **Predicted:** exists e1 e2 x3 x4 x5 x6.(man(x3) & sit(e1) & (subj(e1) = x3) & comfortably(e1) & bench(x4) & on(e1,x4) & woman(x5) & sit(e2) & (subj(e2) = x5) & comfortably(e2))
Prepositional Phrases (PP)	**Sentence:** There is no dog excitedly playing with water [$_{\text{PP}}$ in the grass.]. (sick_test_782_h) **Gold:** -exists e1 x2 x3 x4.(dog(x2) & play(e1) & (subj(e1) = x2) & water(x3) & with(e1,x3) & grass(x4) & in(e1,x4) & excitedly(e1)) **Predicted:** -exists e1 x2 x3 x4.(dog(x2) & play(e1) & (subj(e1) = x2) & water(x3) & with(e1,x3) & grass(x4) & in(e1,x4))
Passive Voice (PSS)	**Sentence:** A rock is being [$_{\text{PSS}}$ climbed] by a person with a rope, which is pink. (sick_test_642_h) **Gold:** exists e1 e2 x3 x4 x5.(rock(x3) & climb(e1) & (obj(e1) = x3) & person(x4) & rope(x5) & pink(x5) & with(e2,x5) & (subj(e2) = x4) & (subj(e1) = x4)) **Predicted:** exists e1 x2 x3 x4.(rock(x2) & climb(e1) & (obj(e1) = x2) & person(x3) & rope(x4) & pink(x4) & with(e1,x4))

Table 8: Examples of Quantifier Count errors in predictions.

ID	Content
sick_test_230_h	**Sentence:** There are no children playing and waiting. **Gold:** -exists e1 e2 x3.(child(x3) & play(e1) & (subj(e1) = x3) & wait(e2) & (subj(e2) = x3)) **Predicted:** -exists e1 x2 x3.(child(x2) & play(e1) & (subj(e1) = x2) & wait(e1))
sick_test_2872_p	**Sentence:** A man is riding a mechanical bull. **Gold:** exists e1 x2 x3.(man(x2) & bull(x3) & mechanical(x3) & ride(e1) & (subj(e1) = x2) & (obj(e1) = x3)) **Predicted:** exists e1 x2 x3 x4.(man(x2) & bull(x3) & mechanical(x4) & ride(e1) & (subj(e1) = x2) & (obj(e1) = x3))