Natural Language Inference with CCG Parser and Automated Theorem Prover for DTS

Asa Tomita and Mai Matsubara and Hinari Daido* and Daisuke Bekki
Ochanomizu University
{tomita.asa, matsubara.mai, hinari.daido, bekki}@is.ocha.ac.jp

Abstract

We propose a natural language inference (NLI) system that operates on the principles of compositional semantics. The system integrates *lightblue*, a syntactic and semantic parser grounded in Combinatory Categorial Grammar (CCG) and Dependent Type Semantics (DTS), with Wani, an automated theorem prover for Dependent Type Theory (DTT). A key feature of this system is that each computational step corresponds to a specific theoretical assumption, allowing the system's evaluation to function as a form of hypothesis verification. We evaluate our inference system using the Japanese Semantic Test Suite (JSeM) and demonstrate how error analyses can provide feedback for refining both the system and its underlying linguistic theory.

1 Introduction

With the advancement of Natural Language Processing (NLP), the gap between formal linguistics and computational linguistics has been widening. Historically, computational linguistics was deeply intertwined with theoretical linguistics, serving as a means to implement and empirically verify formal linguistic theories. However, the field's focus has progressively shifted towards engineering-oriented approaches, a trend significantly accelerated by the rise of large language models (LLMs).

While LLMs have achieved impressive performance on a wide range of NLP tasks, including natural language inference (NLI) (Cobbe et al., 2021; Wei et al., 2022), their reasoning processes are largely *associative* rather than formally grounded. As a result, their inferences are not based on whether a hypothesis is formally deduced from given premises. Although their outputs are often plausible, concerns persist regarding the reliability and explainability of their inferential processes.

In contrast, inference systems based on formal linguistic theories (Bos, 2008; Chatzikyriakidis and Luo, 2014; Abzianidze, 2017) can output formal proof diagrams that explicitly detail the steps of syntactic, semantic, and theorem proving analysis. A notable example is `ccg2lambda` (Mineshima et al., 2015; Martínez Gómez et al., 2016), an inference system the syntactic parser of which employs Combinatory Categorial Grammar (CCG; Steedman, 1996, 2000), a lexicalized grammar that associates syntactic and semantic information with lexical entries. It generates higher-order logical forms, which are then processed by the Coq theorem prover (The Coq Development Team, 2021).

Despite its theoretical foundation, `ccg2lambda` has limitations stemming from its bi-LSTM-based syntactic parser (Yoshikawa et al., 2017). As CCG concentrates linguistic information within the lexicon, neural parsers make it difficult to precisely diagnose errors at the lexical level. In practice, correcting parsing errors often requires modifying the treebank and retraining the model, which hinders its utility for empirical theory verification.

We conceptualize inference as the ability to formally deduce the semantic representation of a hypothesis from that of the premises. To address the aforementioned challenge, we propose an inference pipeline (Figure 1) that combines *lightblue* (Bekki and Kawazoe, 2016), a robust syntactic and semantic parser based on CCG and Dependent Type Semantics (DTS; Bekki, 2014; Bekki and Mineshima, 2017), with Wani (Daido and Bekki, 2017), an automated theorem prover for DTS. We evaluate this pipeline along with a detailed error analysis.

2 Theoretical Background

2.1 Combinatory Categorial Grammar (CCG)

CCG is a lexicalized grammar that models syntactic structures through a lexicon and a set of combinatory rules. We adopt CCG as our syntac-

*This work was conducted independently and does not reflect the views or positions of Amazon Web Services.

Proceedings of the Second Workshop on the Bridges and Gaps between Formal and Computational Linguistics (BriGap-2), pages 1–7

tic framework, because it allows for the explicit encoding of syntactic and semantic information within lexical items, providing a clear and localized representation of linguistic structure. This design is particularly well-suited for computational implementations aimed at the empirical verification of linguistic theories, as parsing errors can often be directly attributed to specific lexical entries, facilitating targeted revision.

2.2 Dependent Type Semantics (DTS)

DTS is a type-theoretical framework for natural language semantics based on Dependent Type Theory (DTT; Martin-Löf, 1984). In DTT, the Curry–Howard correspondence establishes an isomorphism between types as propositions, and between terms as proofs. A key feature of this system is its ability to allow types (propositions) to be dependent on terms (proofs). This property allows DTS to represent propositions (types) that contain a reference to a proof from a preceding discourse, Consequently, it reduces phenomena such as anaphora and presupposition resolution to proof search. Since this proof search mechanism is used to validate inferences from premises to conclusions, DTS provides a unified, proof-theoretic account of meaning. Beyond its handling of anaphora and presupposition resolution, the type-theoretical foundation of DTS also allows for the use of type-checking to ensure the consistency of semantic representations. We will examine this property in detail in Section 3.1.2.

3 Inference Pipeline

3.1 Syntactic/Semantic Parser lightblue

lightblue[1] is a syntactic and semantic parser that integrates CCG-based syntactic parsing with DTS-based semantic composition. The syntactic parsing is grounded in the formalization of Japanese CCG as described in Bekki (2010), and it is capable of generating syntactic structures enriched with detailed syntactic features. Furthermore, lightblue incorporates the anaphora resolution mechanism based on type inference to identify anaphoric relations within discourse. The system also verifies the consistency of the derived semantic representations by performing type checking (Bekki and Sato, 2015).

[1] https://github.com/DaisukeBekki/lightblue

3.1.1 Anaphora and Presupposition Resolution with lightblue

In DTS, type checking is employed to verify whether a semantic representation, obtained through semantic composition, is of type `type` in DTT. This condition is referred to as the Semantic Felicity Condition (SFC). Consequently, this process retrieves the contexts that are available for resolving anaphora and presuppositions.

Pronouns and presupposition triggers introduce underspecified terms into the semantic representations. Following semantic composition, lightblue performs type checking, in which each underspecified type launches a proof search, and the Wani system calculates a corresponding (possibly empty) set of proof terms.

The proof terms derived from the above process are used to rewrite underspecified terms, resulting in fully specified semantic representations. This is the first system to implement anaphora and presupposition resolution in DTS, according to its theoretical formulation. This implementation was made possible by the seamless integration of lightblue and Wani.

3.1.2 Type Checking for Evaluating Semantic Analysis

In CCG, semantic composition is derived from the syntactic structure through a homomorphic mapping. Consequently, any ill-formedness in the resulting semantic representations indicates an inconsistency in the corresponding lexical entries. Therefore, the failure of the SFC, as described in the previous section, directly points to an error in some semantic representation specified in the lexicon. In this way, type checking serves as a valuable tool for verifying the internal consistency of the overall implementation.

3.2 Automated Theorem Prover Wani

Wani (Daido and Bekki, 2017) is an automated theorem prover designed for a specific fragment of DTS. Given a set of premises and a conclusion, both formulated as propositions in DTT, Wani attempts to construct a proof. If a proof is found, it outputs the corresponding DTT proof diagram.

Wani performs proof search by applying DTT inference rules to the premises and the conclusion. It combines forward reasoning and backward reasoning strategies. Forward reasoning proceeds from the premises, applying elimination rules to derive

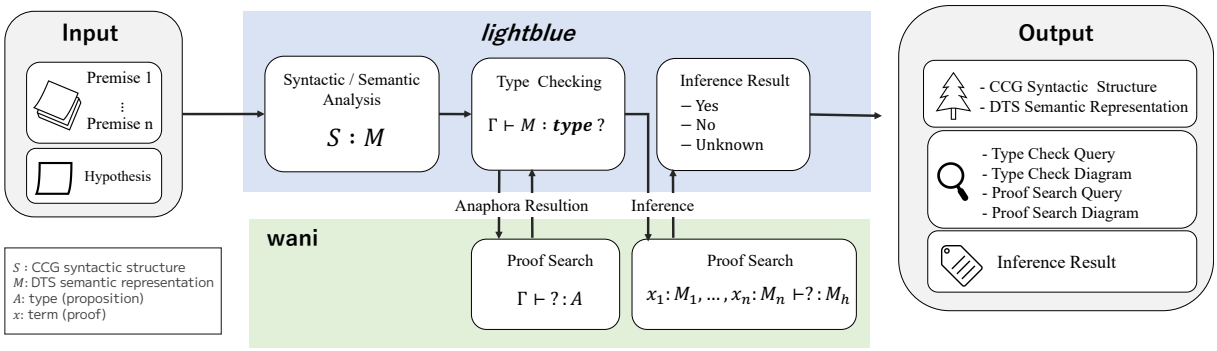

Figure 1: NLI pipeline with lightblue and Wani

their consequences and expand the set of available propositions. In contrast, backward reasoning starts from the conclusion, iteratively working backward to identify the propositions required to apply the rules that would derive it. Wani implements backward reasoning as a depth-first search.

Proof search in DTT is known to be undecidable. To ensure the practical feasibility of Wani, we introduced the following constraints on the search process:

Time and Depth Limits To prevent nontermination, we implemented upper bounds on both the computation time and the number of backward inference steps (i.e., depth). The search is terminated if either of these thresholds is exceeded.

Forward vs. Backward Reasoning While forward reasoning is less flexible, it guarantees termination for elimination rules. Conversely, backward reasoning, while more versatile, can lead to nontermination. To balance these trade-offs, Wani uses forward reasoning for the elimination rule of Σ types and for both the introduction and elimination rules of identity types. All other inference rules are handled via backward reasoning.

Pruning We applied branch pruning to specific backward inference rules for to enhance search efficiency.

3.3 Pipeline Design

The pipeline of the natural language inference system, which utilize lightblue and Wani, is depicted in Figure 1. The process take a set of n-premise sentences and one hypothesis sentence as input, which are then passed to lightblue. For each sentence, lightblue performs syntactic and semantic analyses to compose a semantic representation. A subsequent type-checking procedure is then applied to each semantic representation to ensure that it has the type `type`. At this stage, a context for anaphora and presupposition resolution is incrementally constructed by sequentially adding previously type-checked (fully-specified) semantic representations. This allows type-checking to serve as both a consistency check and a mechanism for anaphora and presupposition resolution. Once all sentences have successfully passed the type-checking phase, Wani is called upon to conduct a proof search. During the search, Wani attempts to construct a proof term of type M_h (the semantic representation of the hypothesis) from the semantic representations of the premise sentences $M_1, \ldots, M_n$. If a proof term is found, Wani returns a proof diagram as output. Based on the output from Wani, lightblue assigns one of three inference labels:

`yes`: A proof term of type M_h is constructed (i.e., the hypothesis is entailed)

`no`: A proof term of type $\neg M_h$ is constructed (i.e., a contradiction)

`unknown`: No proof term is constructed.

Finally, lightblue provides a structured output containing the following information:

Syntactic Structures / Semantic Representations The CCG syntactic structures and DTS semantic composition for the premises $T_1, \ldots, T_n$ and the hypothesis H

Type Checking Information Type checking queries and the corresponding proof diagrams

Proof Search Information Proof search queries and the corresponding proof diagrams

Inference Result The inference label assigned by the system.

JSeM ID:693, Answer:Yes
Premise
ITEL-wa 1993-nen-ni MTALK-o tsukut-ta.
ITEL-TOP 1993-year-in MTALK-ACC
make-PST
(ITEL made MTALK in 1993.)
Hypothesis
ITEL-wa 1993-nen-ni MTALK-o tsukuri-oe-ta.
ITEL-TOP 1993-year-in MTALK-ACC
make-finish-PST
(ITEL finished making MTALK in 1993.)
JSeM ID:703, Answer:Unknown
Premise
Taro-ga Hanako-o sikat-ta.
Taro-NOM Hanako-ACC scold-PST
(Taro scolded Hanako.)
Hypothesis
Taro-ga sikara-re-ta.
Taro-NOM scold-PASSIVE-PST
(Taro was scolded.)

Table 1: Examples in the JSeM dataset

4 Evaluation Experiment

4.1 Dataset: JSeM

The evaluation was conducted on the JSeM dataset (Kawazoe et al., 2015)[2], an inference dataset for Japanese. The dataset contains a mixed set of inference problems: some are direct translations of the English FraCaS test suite (Cooper et al., 1996), while others are specifically designed to address semantic phenomena unique to Japanese. Each problem consists of a set of premises, a hypothesis, and an inference label (yes, no, unknown, or undef, which denotes unacceptable sentences). The problems are further organized into sections categorized in accordance with linguistic phenomena. Examples of data labeled as yes and unknown are shown in Table 1.

4.2 Experiment Setup

We evaluate our system on the 36 problems from the "Verbs" section of JSeM dataset (see Table 1 for examples). This section was selected as it represents the most basic subset of inference problems.

We report the following evaluation metrics: parsing success rate, type-checking success rate, and overall accuracy, precision, recall, and F1 scores. The parsing success rate measures the proportion of problems for which a full syntactic and semantic parse was successfully obtained, as this is a prerequisite for inference. The type checking success rate measures the number of cases where the semantic analysis yielded a well-formed and internally consistent semantic representation. Macro averages treat each class equally, while weighted averages reflect the actual label distribution. Given the class imbalance in the dataset, we report both macro- and weighted-averaged scores for a balanced evaluation.

We emphasize that the 36-problem evaluation set was not used for any system tuning. All components, including parsing, semantic composition, and inference, were applied uniformly without task-specific adjustments.

4.3 Result

Results are shown in Table 2. The evaluation set comprises 72 sentences (36 premises and 36 hypotheses), lightblue generated full parsed trees in 65 sentences, achieving a parsing success rate of approximately 90%. When restricting the evaluation to the 52 unique sentences by removing duplicates, the system achieved full parsed trees for 48, corresponding to a 92.3% success rate. Moreover, type checking succeeded for all parsed sentences, indicating that semantic representations obtained from our semantic analysis satisfied the Semantic Felicity Condition (SFC) and were well-formed. The inference component correctly answered 24 out of the 36 problems. Compared to `ccg2lambda`, our system demonstrated superior performance across all evaluation metrics: accuracy, recall, precision, and F1 score.

Although GPT-4o achieves the highest scores on all metrics, these results should be interpreted as reference values rather than a direct comparison. This is because our research aims at transparent and linguistically grounded inference, which contrasts with the black-box nature GPT-4o. In our framework, a prediction is considered correct only if the system can successfully parse the input, assign a consistent semantic representation, and construct a formal proof. From this perspective, predictions made without a derivable proof, such as GPT-4o's "yes" without an explicit reasoning trace, cannot be fully trusted as valid inferences. Thus, our system prioritizes explainability and credibility based on evidence, over mere surface-level agreement with the correct label.

[2] https://github.com/DaisukeBekki/JSeM

System	ccg2lambda	**Our System**	GPT-4o	Majority
Parsing	-	0.90	-	-
Type Check	-	1.0	-	-
Accuracy	0.556	0.667	0.861	0.806
Precision (macro | weighted)	0.250 | 0.806	0.342 | 0.877	0.438 | 0.951	0.201 | 0.806
Recall (macro | weighted)	0.172 | 0.556	0.397 | 0.667	0.349 | 0.861	0.250 | 1.000
F1 (macro | weighted)	0.204 | 0.658	0.319 | 0.700	0.382 | 0.897	0.223 | 0.892

Table 2: Performance comparison with other systems. Among `ccg2lambda` and our system, the higher value for each metric is underlined. The "Majority" baseline, which assignes the most frequent label ("yes") to all the problems, is also included for reference. For GPT-4o model, we set the temperature to 0.7 and the maximum token limit to 1000. The confusion matrix and the precise prompt used for inference are shown in Table 3 and Figure 2 in the Appendix.

4.4 Error Analysis

Out of the 12 problems with incorrect answers, 7 were attributed to the lack of external world knowledge, 2 to current limitations in Wani's proof search, and the remaining 3 to parsing errors.

4.4.1 External world knowledge

An example of an error attributed to a lack of world knowledge is the following problem:[3]

P: ITEL owned APCOM from 1988 to 1992.

H: ITEL owned APCOM in 1990.

To Correctly infer the hypothesis from the premise, the system requires temporal world knowledge – that 1990 falls within the range from 1988 to 1992 – which is not explicitly encoded.

Incorporating external knowledge presents a well-known challenge. While several studies have explored integrating knowledge bases into their inference systems (Martínez-Gómez et al., 2017; Yoshikawa et al., 2019), these approaches often involve a trade-off where improving recall can lead to a decrease in precision. Therefore, simply injecting more knowledge into the system is insufficient to increase the number of provable cases.

4.4.2 Parsing Error

In our system, we observed parsing errors related to the interpretation of case marks. A representative example is the following sentence[4]:

P: Taro-wa Jiro-kara Hanako-o
Taro-NOM Jiro-from Hanako-ACC
syookaisa -re -ta
introduce PASSIVE PST
'Taro was introduced to Hanako by Jiro.'

In this case, the parser failed to correctly recognize that *kara* ("from") in the passive construction semantically corresponds to the dative argument in the active counterpart. It is known that *kara*-NP is not fully interchangeable with the dative NP, and is not always licensed as a verbal argument. Consequently, resolving such errors requires a deeper linguistic analysis of selectional restrictions and case-marking behavior for specific verbs, rather than a simple modification or addition of lexical entries for *kara*.

5 Conclusion

This paper has presented a linguistically-grounded natural language inference system, which integrates syntactic parsing, semantic composition, type checking, and proof search. Our proposed pipeline demonstrated improved inference accuracy over existing formal systems.

This system offers a promising avenue for bridging the gap between linguistic theory and large language models. Given that all of its technical components are based on hypotheses from formal linguistics, improvements to the system directly contribute to the refinement of theoretical assumptions. Furthermore, lightblue can serve as a novel tool for verifying the outputs of LLMs, thereby facilitating systematic comparisons between data-driven inferences and theory-driven predictions.

Acknowledgments

This work was supported by JST BOOST, Japan Grant Number JPMJBS2406, JSPS KAKENHI Grant Number JP23H03452, Japan, and JST CREST Grant Number JPMJCR20D2, Japan.

[3]JSeM ID: #698
[4]JSeM ID: #717

References

Lasha Abzianidze. 2017. LangPro: Natural language theorem prover. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 115–120, Copenhagen, Denmark. Association for Computational Linguistics.

Daisuke Bekki. 2010. *Nihongo-Bunpoo-no Keisiki-Riron - Katuyootaikei, Toogohantyuu, Imigoosei - (trans. 'Formal Japanese Grammar: the conjugation system, categorial syntax, and compositional semantics')*. Kuroshio Publisher, Tokyo.

Daisuke Bekki. 2014. Representing anaphora with dependent types. In *Logical Aspects of Computational Linguistics*, pages 14–29, Berlin, Heidelberg. Springer Berlin Heidelberg.

Daisuke Bekki and Ai Kawazoe. 2016. Implementing variable vectors in a CCG parser. In *Logical Aspects of Computational Linguistics. Celebrating 20 Years of LACL (1996–2016)*, pages 52–67, Berlin, Heidelberg. Springer Berlin Heidelberg.

Daisuke Bekki and Koji Mineshima. 2017. *Context-Passing and Underspecification in Dependent Type Semantics*, pages 11–41. Springer International Publishing, Cham.

Daisuke Bekki and Miho Sato. 2015. Calculating projections via type checking. In *TYpe Theory and LExical Semantics (TYTLES), ESSLLI2015 workshop*.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, pages 277–286. College Publications.

Stergios Chatzikyriakidis and Zhaohui Luo. 2014. Natural language inference in Coq. *Journal of Logic, Language and Information*, 23(4):441–480.

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

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Josef Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, Steve Pulman, Ted Briscoe, Holger Maier, and Karsten Konrad. 1996. FraCaS: A Framework for Computational Semantics. Technical Report Deliverable D16, FraCaS Consortium.

Hinari Daido and Daisuke Bekki. 2017. Development of an automated theorem prover for the fragment of DTS. In *the 17th International Workshop on Logic and Engineering of Natural Language Semantics (LENLS17)*.

Ai Kawazoe, Ribeka Tanaka, Koji Mineshima, and Daisuke Bekki. 2015. A framework for constructing multilingual inference problem sets: Highlighting similarities and differences in semantic phenomena between English and Japanese. In *MLKRep2015*.

Per Martin-Löf. 1984. *Intuitionistic Type Theory Vol. 1*. Bibliopolis.

Pascual Martínez Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: A computational semantics system. In *the Association of Computational Linguistics (ACL2016)*, 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, Lisbon, Portugal. Association for Computational Linguistics.

Mark Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press, Cambridge.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

The Coq Development Team. 2021. The coq reference manual: Release 8.14.1.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Masashi Yoshikawa, Koji Mineshima, Hiroshi Noji, and Daisuke Bekki. 2019. Combining axiom injection and knowledge base completion for efficient natural language inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7410–7417.

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 Appendix

		GPT 4o				ccg2lambda				lightblue			
		Yes	No	Unk	Other	Yes	No	Unk	Other	Yes	No	Unk	Other
Ground Truth	Yes	28	0	1	0	20	0	1	8	17	0	12	0
	No	0	0	0	0	0	0	0	0	0	0	0	0
	Unk	0	4	3	0	0	0	0	7	0	0	7	0
	Other	0	0	0	0	0	0	0	0	0	0	0	0

Table 3: Confusion matrix of inference systems

与えられた文のペアについて、文Aが文Bを含意するかどうかを判定し、そのペアに⟨DATASET_LABEL⟩を付けてください。⟨DATASET_LABEL⟩は以下のいずれかです：
yes：前提が仮説を含意する
no：前提が仮説の否定を含意する
unknown：前提が仮説を含意せず、その否定も含意しない
undef：与えられた情報のみからは判断ができない

文A：⟨PREMISE_SENTENCE⟩
文B：⟨HYPOTHESIS_SENTENCE⟩
####
⟨DATASET_LABEL⟩のみ出力してください。

—- English Translation —-
Given a pair of sentences, determine whether Sentence A entails Sentence B, and assign a ⟨DATASET_LABEL⟩ to the pair.⟨DATASET_LABEL⟩ must be one of the following:
yes: the premise entails the hypothesis
no: the premise entails the negation of the hypothesis
unknown: the premise entails neither the hypothesis nor its negation
undef: it is not possible to determine based on the given information alone

Sentence A: ⟨PREMISE_SENTENCE⟩
Sentence B: ⟨HYPOTHESIS_SENTENCE⟩

####
Only output ⟨DATASET_LABEL⟩.

Figure 2: Prompt designed for LLMs to assign the entailment relation label <DATASET_LABEL>, and its English translation