Yesod による日本語 CCG パーザ開発環境の構築

石嶋美咲　(指導教員：戸次 大介)

1　はじめに

まず、背景として CCG（Combinatory Categorial Grammar, 組み合わせ範疇文法）[1] パーザの文法開発は一般的に高価である。その理由は、主に二点挙げられる。一点目は、実テキストに文法規則を適用できる人材が少ない点。二点目は、木構造の整合性のチェックは、線形構造の整合性、例えば品詞の割り当て等のチェックと比べて高価である点である。

このことから、GUI（Graphical User Interface）を備えた、文法の知識を持ち合わせ、かつ、様々な機能がついたシステムを制作したいと考えた。本研究では日本語 CCG パーザ lightblue[2] と Web フレームワーク Yesod[3] を用いて目的のシステムを実装していく。本研究で作成するシステムでは主に下記の 2 点を実現させたいと考えている。

- CCG 木構造の可視化に留まらず、それを編集する機能や視認性を上げる機能を装備する
- 編集された CCG 木構造を CCG ツリーバンク形式で保存・上書きする

本研究の最終的な目標は、日本語 CCG の文法開発環境を提供することである。文法開発環境とは、パーザとツリーバンクを相補的に改良することが可能な環境である。将来的には、本研究で作成した GUI で編集・保存した CCG ツリーバンクを学習データとして、ligtblue などのパーザの訓練に用いたいと考えている。本稿では、上記で挙げたうち CCG 木構造の視認性を上げるパーザ開発環境“ express ”について主に述べる。

2　先行研究

GUI を持つパーザ開発プロジェクトとして主に下記の 4 つが挙げられる。

- PMB (The Parallel Meaning Bank)[4]
- Stanford CoreNLP[5]
- Allen NLP[6]
- NPCMJ (NINJAL Parsed Corpus of Modern Japanese)[1]

これらの先行研究の共通点として、構文木の可視化が可能であるという点が挙げられる。表 1 は先行研究において実現されている機能を比較したものである。本研究では表 1 にある全項目を満たすシステムを作成することを目指している。その上で先行研究と本研究との差分を列挙していく。

最初に、本研究において最も重要視する項目は、日本語に対応しているかどうかである。その中で

[1]国立国語研究所 (2018)『NPCMJ Explorer』（http://npcmj.ninjal.ac.jp/explorer/）

プロジェクト	日本語	ツリーバンク編集	意味表示	CCG
PMB	△	○	○	○
Stanford CoreNLP	×	×	△	×
Allen NLP	×	×	△	×
NPCMJ	○	×	○	×

表 1: 本研究と先行研究の差分

PMB は日本語への対応が開始されたばかりであるため、△となっている。次に、ツリーバンクの編集については、品詞の変更などが例として挙げられる。PMB はツリーバンクの編集の機能が非常に充実している。意味表示については、意味役割の付与にとどまるものは△、論理式で表現されているものは○とした。最後は、採用している文法理論についてである。現在、CCG を採用しているのは唯一 PMB だけである。

3　システムの概要

このシステムでは入力を日本語の文とし、その入力文に対する CCG 木構造と意味表示をブラウザ上に出力する。そして、ユーザーが出力された CCG 木に対して編集できるような機能を装備することを目指している。このセクションでは、システムの大きな構成要素である lightblue と Yesod について述べる。システムの概要は図 1(次項) に示す。

3.1　lightblue

lightblue は日本語 CCG パーザである。入力は日本語の文とし、それに対する CCG 木構造と意味表示が出力される。意味表示は DTS（Dependent Type Semantics, 依存型意味論）[7] を用いている。lightblue は純粋関数型プログラミング言語である Haskell を用いて実装されている。

3.2　Yesod

Yesod について簡潔に説明すると、Haskell で書かれた Web アプリケーションフレームワークである。Yesod では標準的な Haskell を拡張した文法である TemplateHaskell[8] が使用可能であり、その中で Widget などの動的なコンテンツが生成できる。上記で述べたように lightblue は Haskell で実装されているため、Widget の中で Haskell コードが呼び出せることは Yesod を用いる利点の 1 つである。このシステムにおいて、ユーザーの操作部分とそれに伴う画面表示部分を Yesod で対応する。

現段階では、JSeM（Japanese Semantic test suite：日本語意味論テストセット）[9] に対する CCG 木構造の出力が可能である。JSeM とは、日本語の意味的な現象に基づく含意関係のデータセットを構築したものである。そのテストセットは、FraCas test suite[10] の方針にならい言語現象ごとに含意関係のテストをまとめている。テス

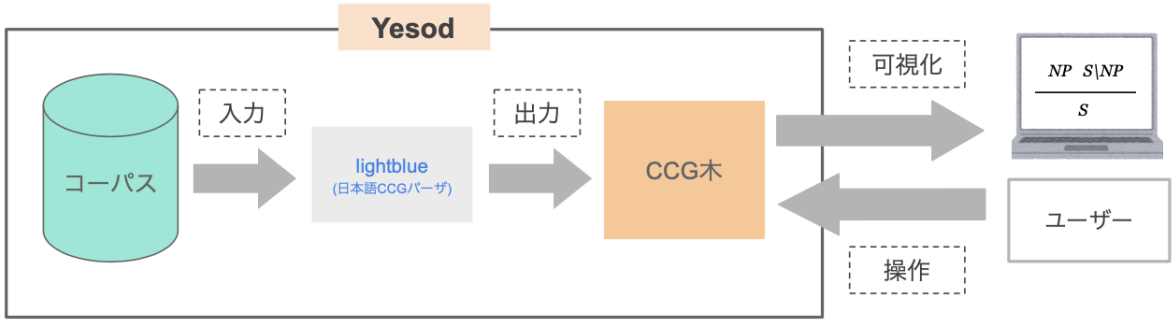

図 1: システムの概要

トは主に、前提 (premises)・仮説 (hypothesis)・判定 (answer) から構成されている。前提は 1 つ以上の文、仮説は 1 つの文となっており、判定は前提と仮説の間の含意関係により yes, no, unknown もしくは undef のいずれかのラベルが与えられている。現在のシステムでは、その JSeM のテストセットの指定された jsem_id の前提と仮説の文それぞれに対する CCG 木構造を出力することができる。(図 1)

4 まとめ

本稿では、日本語 CCG パーザである lightblue と Web アプリケーションフレームワークである Yesod を用いて文法開発システムを実装した。木構造を可視化した際、木の折り畳み/展開など様々な機能が装備されると、木構造の分析にとって有用である。今後の課題としては、意味表示として用いている DTS をブラウザ上に出力、CCG 文法に沿った品詞の修正、出力された CCG 木の保存などの機能を装備したいと考えている。

参考文献

[1] Steedman M. Surface structure and interpretation. MIT Press, 1996.

[2] Bekki D and Kawazoe A. Implementing variable vectors in a ccg parser. In *In Logical Aspects of Computational Linguistics (9th international conference, LACL2016, Nancy, France, December 2016 Proceedings)*, pp. 52–67, 2016.

[3] Snoyman M. *Developing Web Apps with Haskell and Yesod, Second Edition.* O'Reilly Media, Inc, 2nd edition, 2015.

[4] Abzianidze L, Bjerva J, Evang K, Haagsma H, Noord R, Ludmann P, Nguyen D, and Bos J. 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*, pp. 242–247, 2017.

[5] Manning C, Surdeanu M, Bauer J, Finkel J, Bethard S, and McClosky D. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014.

[6] Joshi V, Peters M, and Hopkins M. Extending a parser to distant domains using a few dozen partially annotated examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 1190–1199, 2018.

[7] Bekki D and Mineshima K. Context-passing and underspecification in dependent type semantics. In *Modern Perspectives in Type-Theoretical Semantics, S.Chatzikyriakidis and Z.Luo (Eds.), Studies of Linguistics and Philosophy, Springer*, pp. 11–41, 2017.

[8] Sheard T and Jones S. Template meta-programming for haskell. In *Proceedings of the 2002 ACM SIGPLAN workshop on Haskell*, pp. 1–16, 2002.

[9] Kawazoe A, Tanaka R, Mineshima K, and Bekki D. An inference problem set for evaluating semantic theories and semantic processing systems for japanese. In *In Proceedings of the Twelfth International Workshop on Logic and Engineering of Natural Language Semantics (LENLS12), JSAI International Symposia on AI 2015*, pp. 67–73, 2015.

[10] Cooper R, Crouch D, van Eijck J, Fox C, van Genabith J, Jan J, Kamp H, Milward D, Pinkal M, Poesio M, Pulman S, Briscoe T, Maier H, and Konrad K. Using the framework. In *Technical report,FraCaS: A Framework for Computational Semantics. FraCaS deliverable D16.*, 1996.