Interactive CCG Parsing with Incremental Trees*

Masaya Taniguchi[1], Satoshi Tojo[1], and Koji Mineshima[2]

[1]Japan Advanced Institute of Science and Technology
[2]Keio University

In natural language processing, an ordinary parser constructs a syntactic structure, given the whole sentence; however, we human grasp the meaning of a given sentence word by word sequentially. So, in this work we aim at constructing a partial semantics structure incrementally from the beginning word of a sentence.

The compositionality principle states that the semantics is obtained in parallel to the completion of a syntactic structure. Thus, to obtain a partial semantics we need to assume that there is partial tree structure even in the middle of utterance. To achieve this property, we employ combinatory categorial grammar (CCG) [3], which enables us to acquire a semantic structure and a parsing tree simultaneously. However, those conventional grammar rules in CCG are insufficient for the incremental parsing. Therefore, we introduce new rules as a natural extension of existing formalisms, and in addition, we implement an interactive parser that can externalize a parsing tree for each word input.

For example, we consider obtaining a tree of "Alice" and "Alice loves" from the sentence "Alice loves Bob," found in the incremental parsing of the sentence. Then, we construct a partial tree whenever a new succeeding word appears, regarding it as a token consumption. As a result, we find a left-branching tree in Figure 1, without employing extra memory for floating tokens that are un-treed words.

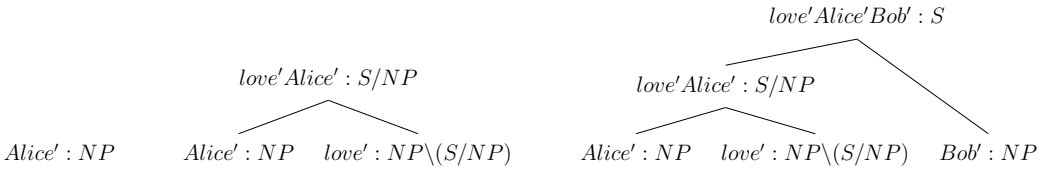

Figure 1: Transition of the left-branching tree

From the semantic point of view, we need to preserve the semantics, that is a sequence of terms obtained from lexical items, between the non-left-branching tree and the left-branching tree if they consist of the same categories. This is because categorial grammar and its derivations have a confluence property in the simply-typed lambda calculus; that is, the reduction steps do not affect the resultant sequence of semantics terms.

From the syntactic point of view, we can generally assign multiple different categories for each word because these categories always remain equivocal until the end of the sentence. However, we do not argue this issue in this paper; instead, we deal with all the possible category assignments.

*This work was supported by Grant-in-Aid for JSPS Fellows Number 21J15207.

Stanjević and Steedman [2] implemented an incremental parsing algorithm employing the tree rotation which depends on the following composition rules.

$$X/Y \quad W\backslash(Y/Z) \Rightarrow_{<B^2_x} W\backslash(X/Z) \qquad W\backslash(Y/Z) \quad Y\backslash X \Rightarrow_{>B^2} W\backslash(X/Z)$$

As the first contribution, we argue that the introduction of B^2 and B^2_x in [2] are rather haphazard and are not linguistically indispensable. In general, it is not preferable to introduce new rules to solve certain specific language phenomena. If we admit the introduction of B^2, B^2_x or possible introduction of B^3 mentioned in [2], we may also need to consider B^4 or more B^n. Considering the possibility that we may face such a situation, we should provide a general principle to tolerate any number of arguments for a given word. This is the reason for which we would like to reduce these binary, or n-ary operations to unary derivation rules.

We here get back to the principles of CCG, and introduce a set of derivation rules of D, Dx, Q, Qx.

$$\begin{array}{lll} X/Y \Rightarrow_{>D} (X/Z)/(Y/Z) & X/Y \Rightarrow_{>Q} (Z/X)\backslash(Z/Y) & X\backslash Y \Rightarrow_{<D} (Z\backslash X)\backslash(Z\backslash Y) \\ X\backslash Y \Rightarrow_{<Q} (X\backslash Z)/(Y\backslash Z) & X/Y \Rightarrow_{>Dx} (Z\backslash X)/(Z\backslash Y) & X\backslash Y \Rightarrow_{>Qx} (Z/Y)\backslash(X\backslash Z) \\ X\backslash Y \Rightarrow_{<Dx} (X/Z)\backslash(Y/Z) & X/Y \Rightarrow_{<Qx} (Z/Y)/(X\backslash Z) & \end{array}$$

Here are comparisons;

- B^2 and B^2_x correspond to two operations; one is the composition with two arguments and another is to indicate the head of the constituent.
- D, Dx, Q, and Qx correspond to the single operation that is to indicate the head of the constituent as the same as the type-raising rules, *e.g.*, a pair of categories $X/Y, Y$ is to be $X/Y, (X/Y)\backslash X$ in a derivation, by which we switch the argument part to the functional part, and so do the functional part to the argument part.

Note that while we can derive B^2 and D^2_x from D, Dx, Q, and Qx, the reverse does not hold. Therefore, our extension is a generalization of the preceding work [2]. Furthermore, our new rules are unary so that the derivation is more versatile and can derive a certain set of sentences that are not obtained only by binary rules. Instead, the derivation steps may increase when we employ only unary rules.

As the second contribution, we implement our CCG parser in logic programming. The main predicate takes three arguments as in `parse(Grammar, Sentence, Tree)`, namely, a set of grammar rules, a sentence, and a tree. Generally, we build a constituency tree structure from a sentence using the parser with some given grammar rules, *i.e.*, the building process is a sequence of unification between applicable `Grammar` rules and a given `Sentence`. However, we can execute the parser in the reversal way, as in definite clause grammar [1]; when we execute the unification between an input `Sentence` and a parsed `Tree`, we can replicate a set of innate `Grammar` rules.

Based on these two principles, we implement an interactive and incremental CCG parser. This system takes a sentence as input and displays the possible parsing results in CCG. For each word input, the partial sentence is processed as follows;

1. A syntax tree is created for the partial sentence using Stanford CoreNLP. This result is shown in the left-bottom corner of Figure 2.
2. The tree and the sentence are used to extract the grammar rules employed in the parser. This result is shown in the left-top corner of Figure 2.
3. Based on the extracted grammar, the parser outputs all possible parsing results in CCG. This result is shown in the right-hand side of Figure 2.

In the grammar extraction stage, the part-of-speech (POS) is regarded as a variable for logic programming, and grammar rules are extracted by the process of unification. For this reason, as mentioned above, the type of the partial tree is not uniquely determined, and so all the possible grammar rules are exhaustively consulted. With this system, we illustrate the growing process of a left-branching tree. Moreover, it also shows how the parser builds a semantics structure from a sentence.

Our future extension includes the reduction of computational complexity; the search by Prolog is non-deterministic and exhaustive, so that the computation is not efficient. We need to consider effective pruning in the bidirectional search, to be applicable to practical sentence length.

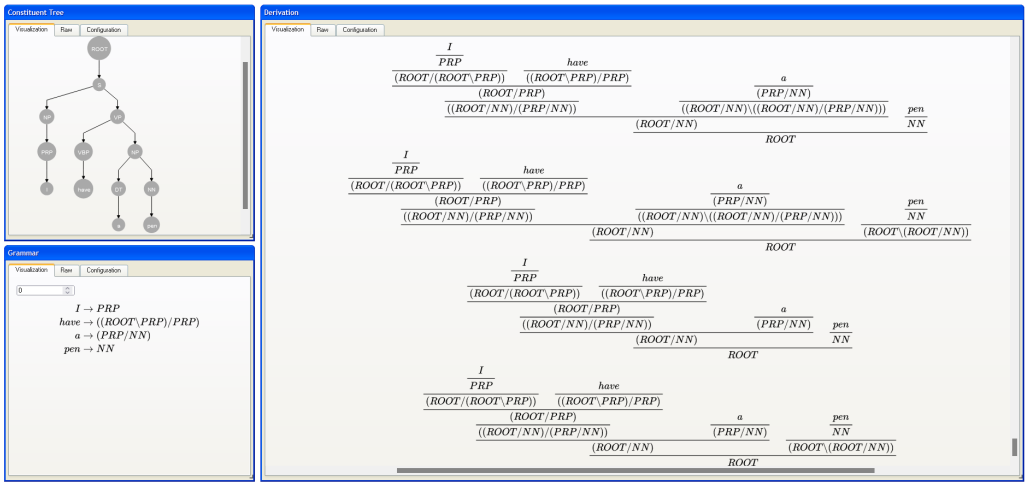

Figure 2: Interactive CCG Parser

Table 1: Extension of CCG Rules

$X/Y \quad Y \Rightarrow_{>} X$	$Y \Rightarrow_{>T} (X/Y)\backslash X$		
$X \quad X\backslash Y \Rightarrow_{<} Y$	$X \Rightarrow_{<T} Y/(X\backslash Y)$		
$X/Y \quad Y/Z \Rightarrow_{>B} X/Z$	$X/Y \Rightarrow_{>D} (X/Z)/(Y/Z)$	$X/Y \Rightarrow_{>Q} (Z/X)\backslash(Z/Y)$	
$X\backslash Y \quad Y\backslash Z \Rightarrow_{<B} X\backslash Z$	$X\backslash Y \Rightarrow_{<D} (Z\backslash X)\backslash(Z\backslash Y)$	$X\backslash Y \Rightarrow_{<Q} (X\backslash Z)/(Y\backslash Z)$	
$X/Y \quad Z\backslash Y \Rightarrow_{>Bx} Z\backslash X$	$X/Y \Rightarrow_{>Dx} (Z\backslash X)/(Z\backslash Y)$	$X\backslash Y \Rightarrow_{>Qx} (Z/Y)\backslash(X\backslash Z)$	
$X/Y \quad X\backslash Z \Rightarrow_{<Bx} Z/Y$	$X\backslash Y \Rightarrow_{<Dx} (X/Z)\backslash(Y/Z)$	$X/Y \Rightarrow_{<Qx} (Z/Y)/(X\backslash Z)$	

References

[1] Fernando Pereira and David Warren. “Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks”. In: *Artificial intelligence* (1980).

[2] Milo Stanojevi and Mark Steedman. “CCG parsing algorithm with incremental tree rotation”. 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)*. 2019, pp. 228–239.

[3] Mark Steedman. *The Syntactic Process*. MIT press, 2001.