証明木作成プログラムを用いたCCG統語導出の実装

尾崎有梨　　櫻井加奈子　　浅井健一　　戸次大介
お茶の水女子大学 理学部 情報科学科
〒 112-8610 東京都文京区大塚 2-1-1

1　はじめに

自然言語の意味構造や統語構造を分析するにあたり、様々な文法理論が存在するが、その中でも、CCG (Combinatory Categorial Grammar:[1][2][3]) は近年、頑健なパーザのための文法理論として注目されている。しかし、自然言語処理における有用性が示される一方で、文法理論と論理・型システムの間の強い対応が失われてきている。自然言語の意味構造や統語構造を論理・型システムと対応付けることは、それぞれの数学的解釈につながるという点で、言語学的に重要である。

そこで本研究は、証明木作成プログラム Mikiβ[5] を用いて、CCG の統語導出を型推論として実装し、その過程を通して CCG の型システムとしての再解釈を試みた。

2 節で CCG の概要を紹介し、3 節に CCG 統語導出の GUI 構築について述べる。4 節では、型システムとしての現在の CCG の問題点を指摘する。

2　CCG の概要

CCG は語彙化文法の一種であり、多数の語彙項目と少数の組み合わせ規則からなる。CCG はその他の文法理論と比べ、等位接続構造の分析等において優れていることが知られている。[1]

2.1　辞書と語彙項目

CCG では、構成素を記号列、意味表示、統語範疇の 3 つ組で表す。意味表示には型付きラムダ計算が用いられるのが通例である。統語範疇は品詞に相当する概念である。辞書においては意味表示と統語範疇が各語に対して割り当てられている (それを語彙項目と呼ぶ)。

例えば、

$$\begin{aligned} eat &\Rightarrow (S\backslash NP)/NP \\ apples &\Rightarrow NP \end{aligned}$$

のように、“eat”、“apples” に統語範疇が定められている。また、各語の統語範疇は、主に名詞は NP, 他動詞は $(S\backslash NP)/NP$ であるが、品詞によって一意に決まっているわけではない。

2.2　組み合わせ規則

語や句からより大きな構成素をつくるには、組み合わせ規則を用いる。CCG では、X/Y という統語範疇をもつ構成素があったとき (X, Y は任意の統語範疇)、その右側に Y という統語範疇をもつ構成素があればふたつは組み合わせることができ、結果 X となるという規則が存在する。これを順関数適用規則 (>) といい以下のように定義されている。

$$>\frac{\Gamma \Rightarrow f : X/Y \quad \Delta \Rightarrow a : Y}{\Gamma, \Delta \Rightarrow fa : X}$$

同様に、$X\backslash Y$ という統語範疇をもつ構成素があったとき、その左側に Y という統語範疇を持つ構成素があればふたつを組み合わせることができ、結果 X になる規則がある。これを逆関数適用規則 (<) といい以下のように定義されている。

$$<\frac{\Gamma \Rightarrow a : Y \quad \Delta \Rightarrow f : X\backslash Y}{\Gamma, \Delta \Rightarrow fa : X}$$

CCG の規則は他にも次のようなものがある。

$$>_B\frac{\Gamma \Rightarrow f : X/Y \quad \Delta \Rightarrow g : Y/Z}{\Gamma, \Delta \Rightarrow \lambda x.f(gx) : X/Z} \qquad <_B\frac{\Gamma \Rightarrow g : Y\backslash Z \quad \Delta \Rightarrow f : X\backslash Y}{\Gamma, \Delta \Rightarrow \lambda x.f(gx) : X\backslash Z}$$

$$>_{S_\times}\frac{\Gamma \Rightarrow f : (X/Y)\backslash Z \quad \Delta \Rightarrow g : Y\backslash Z}{\Gamma, \Delta \Rightarrow \lambda x.fx(gx) : X\backslash Z} \qquad <_{S_\times}\frac{\Gamma \Rightarrow g : Y/Z \quad \Delta \Rightarrow f : (X\backslash Y)/Z}{\Gamma, \Delta \Rightarrow \lambda x.fx(gx) : X/Z}$$

$$>_T\frac{\Gamma \Rightarrow a : X}{\Gamma \Rightarrow \lambda f.fa : \boldsymbol{T}/(\boldsymbol{T}\backslash X)} \qquad <_T\frac{\Gamma \Rightarrow a : X}{\Gamma \Rightarrow \lambda f.fa : \boldsymbol{T}\backslash(\boldsymbol{T}/X)}$$

$$<_\Phi>\frac{\Gamma_1 \Rightarrow f_1 : X \quad \cdots \quad \Gamma_{n-1} \Rightarrow f_{n-1} : X \quad \Delta \Rightarrow \circ : CONJ \quad \Gamma_n \Rightarrow f_n : X}{\Gamma_1, \cdots, \Gamma_{n-1}, \Delta, \Gamma_n \Rightarrow \lambda\vec{x}.(f_1\vec{x}) \circ \cdots \circ (f_n\vec{x}) : X}$$

上から順に、順関数合成規則 ($>B$)、逆関数合成規則 ($<B$)、順型繰り上げ規則 ($>\boldsymbol{T}$)、逆型繰り上げ規則 ($<\boldsymbol{T}$)、順関数交差置換規則 ($>S_\times$)、逆関数交差置換規則 ($<S_\times$)、等位接続規則 ($<\Phi>$) という。X や Y、$\boldsymbol{T}$ は任意の統語範疇である。

例えば、“Keats eats apple” という文の場合の構文解析は、

$$<\frac{Keats \Rightarrow NP \qquad >\dfrac{eats \Rightarrow (S\backslash NP)/NP \quad apples \Rightarrow NP}{eats, apples \Rightarrow S\backslash NP}}{Keats, eats, apples \Rightarrow S}$$

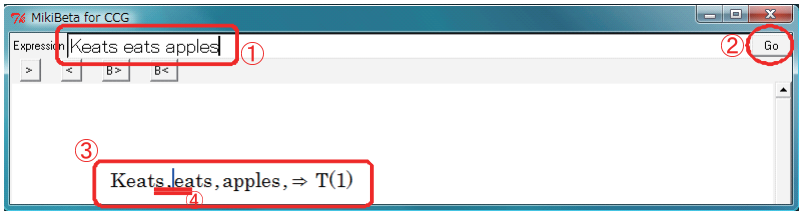

図 1　統語導出動作例 1

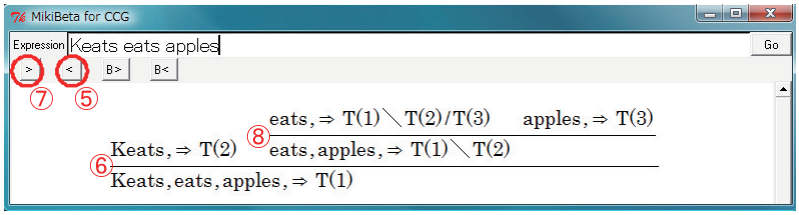

図 2　統語導出動作例 2

となる。(以下、意味表示は省略する。)

まず “eats” と “apples” が順関数適用規則 (>) によって組み合わせられ $S\backslash NP$ となり、今度はそれと”Keats”が逆関数適用規則 (<) によって組み合わせられている。このように、規則のいずれかを構成素の組に適用して文を生成する。

2.3　型システムとの関連

2.1 節で述べたように、CCG における構成素は、記号列、型付きラムダ計算で表された意味表示、統語範疇とで構成されている。一方型システムは、環境・項・型で構成されており、構造的に CCG と型システムには対応関係がある。これにより、組み合わせ規則と型付け規則の間にも本来対応があると考えられる。

この対応関係がどの程度厳密なものであるかという問題は、次の二つの問題に分割される。

1. CCG の記号列・意味表示・統語範疇の全ての情報がプログラミング言語の環境・値 (式)・型として記述されうるか。

2. 1 が可能である場合、CCG の統語範疇の推論を型推論の問題とみなして計算できるか。

本研究では CCG の規則による証明木作成 GUI を実装する過程を通して、この二つの問題を考察する。

3　実装

CCG 統語導出の実装の手法を説明する。

3.1　証明木作成プログラム「Mikiβ」

Mikiβ[5] は、シーケント計算の推論規則を用いた証明木作成 GUI を OCaml の lablTk ライブラリを用いて実装した。本研究ではこの Mikiβ をもとに、シーケント計算の推論規則を CCG の組み合わせ規則に換えて統語導出 GUI を実装した。なお、ここでは意味表示は扱っていない。

3.2　シーケント計算から CCG への変更点

ここからさらに、CCG の規則を組み込むために、統語構造を表す型の型をつくった。型の OCaml による表現と対応する CCG のの統語範疇は以下の表の通りである。

Variable	内容
`T(n)`	基底範疇
`Fun1(X,Y,id)`	X/Y
`Fun2(X,Y,id)`	$X\backslash Y$

X/Y という型の統語範疇だった場合 `Fun1(X,Y,id)` とその統語範疇を構成している統語範疇 X, Y のデータを取れるようにしている。`id` とはノードを識別するために定義されたものでどの型にもついている。

[5] と同様に、証明木の木構造を形成する `Tree` 型の中には、`id` と一緒に、各段に表示させる環境・型の型がある。`Tree` が入れ子構造になることによって証明木が作成される。あるノードの型と別のノードの型が同じかを確認する時 (`Unify`) は、`Unify` 成功後に `id` を共有することによって、内部データを結びつける。

3.3　動作例

実装の動作例をを説明する (図 1、図 2)。

図 1 より、入力欄に解析したい文を入力 (①)、「Go」ボタン (②) をクリックすると、下のフレームに、文が生成された状態の記号列・統語構造が表示される (③)。なお図のように、この段階では具体的な統語範疇は不

明であり、3.2 節で述べた、T(n) で基底範疇を表す。

ここから、文がどのように組み合わせられたかを考える。

(③) の文の部分にポインタを合わせると、語と語の堺に区切り線が現れる (④) ので、“Keats” と “eats apples” で逆関数適用規則を適用させようとした場合は、まず “Keats” と “eats apples” の間に区切り線が出る状態でクリックし、区切る位置を確定する。それから、上にある逆関数適用の記号が書かれたボタン (⑤) をクリックする。すると、(⑥) のように、逆関数適用後の構成素が上段に表示される。型部分も規則適用にしたがっている。“eat apples” も同じように、区切り位置を確定後、順関数適用規則ボタン (⑦) をクリックすると、(⑧) に適用後の構成素が表示される。なお T(n) は Unify 後、統語範疇が決定したら変更するようにする。

4 問題点

4.1 型繰り上げ規則

2.2 節で述べた統語構造規則のなかで、型繰り上げ規則 ($>T$,$<T$) は、本来の統語範疇を変数 $\boldsymbol{T}$ を含む別の統語範疇の形にできる (以下再掲)。

$$>^T\frac{\Gamma \Rightarrow a : X}{\Gamma \Rightarrow \lambda f.fa : \boldsymbol{T}/(\boldsymbol{T}\backslash X)} \qquad <^T\frac{\Gamma \Rightarrow a : X}{\Gamma \Rightarrow \lambda f.fa : \boldsymbol{T}\backslash(\boldsymbol{T}/X)}$$

例えば以下のように、環境が “Keats eats” の場合は、

$$>\frac{>^T\dfrac{Keats \Rightarrow NP}{Keats \Rightarrow \boldsymbol{T}/(\boldsymbol{T}\backslash NP)} \qquad eats \Rightarrow (S\backslash NP)/NP}{Keats, eats \Rightarrow S}$$

のように、“Keats” の型を $\boldsymbol{T}/(\boldsymbol{T}\backslash NP)$ として扱い、順関数合成規則を可能にしている。

この型繰り上げ規則には、シーケント計算や型システムの推論規則にはない特徴がある。それは、下段の統語範疇の中で同じ変数 $\boldsymbol{T}$ が複数回現れていることである。これにより、必然的に「構造共有」が生じる。

型繰り上げ規則をトップダウンに適用する例をみてみる。

$$>^B\frac{>^T\dfrac{Keats \Rightarrow T(4)}{Keats \Rightarrow T(1)/T(3)} \qquad eats \Rightarrow T(3)/T(2)}{Keats, eats \Rightarrow T(1)/T(2)}$$

但し、$T(3)$=$T(1)\backslash T(4)$ —(†)

適用先の “Keats” の統語範疇は $T(1)/T(3)$ である。これを $\boldsymbol{T}/(\boldsymbol{T}\backslash X)$ とみなすと、$T(1)$ と $T(3)$ の内部が同じものを指していることになる。しかし、$T(1)$ の表す統語範疇はこの段階では決定していないため、(†) の制約を保持しなければならない。すなわち、以下のようなデータ構造が必要となる。

T(1)	⟶	○
T(3)	⟶	Fun2(○, ●)
T(4)	⟶	●

現在の Mikiβ の仕様では、型変数の指示先、およびその内部に、別の型変数が現れることはできないため、この例を扱うためにはより一般的な Unify の仕組みが必要である。

5 おわりに

本研究では、[5] に基づき、CCG の統語導出過程を GUI で操作できる証明木作成プログラムを実装した。

それにより、構文解析や文生成の処理とは異なる、型推論の処理の観点からみた場合、CCG では型繰り上げ規則によって生じる構造共有がシーケント計算や型システムとしての推論規則にはない難しさをもたらしていることを指摘した。

現在、この問題を解決するために、メタプログラムの手法、もしくは、Tree 型を導出後用と規則適用時用とに分けてつくり、二つの木の 1 対 1 対応によって Unify を決定するという手法を用いて製作中である。

また、すべての規則について考察できていないので、CCG の規則を全て型システムに対応させることや、三人称単数などの統語素性導入も、今後の課題である。

参考文献

[1] Steedman, Mark. *Surface Structure and Interpretation*, The MIT Press, 1996.

[2] Steedman, Mark. *The Syntactic Process*, The MIT Press, 2001.

[3] Steedman, Mark, and Baldridge, Jason. *Combinatory Categorial Grammar*, In Borsley, R. and Borjars, K. (eds.), *Non-Transformational Syntax*. Blackwell, 2007.

[4] 戸次 大介,「日本語文法の形式理論—活用体系・統語導出・意味合成—」, くろしお出版, 2010.

[5] 櫻井 加奈子, 浅井 健一,「汎用的に証明木を作成する『Mikiβ』」, (投稿中).