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　(指導教員：戸次大介)

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。そのうちの一つである CCG (Combinatory Categorial Grammar:[1][2]) は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]。しかし、以上の理由により CCG の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[3] 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　CCG の概要

2.1　辞書と組み合わせ規則

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。意味表示の記述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の記号列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、

$$eat \Rightarrow (S\backslash NP)/NP$$
$$apples \Rightarrow NP$$

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、このような伝統文法の品詞と一対一に対応しているわけではない。

さらに、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 (>) という。

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

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、X になる規則が存在する。これを逆関数適用規則 (<) という。

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

その他にも 7 つの規則があり、これらの規則を適用することによって文を生成する。

例えば、“Keats eats apple” という文の場合の導出は、

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

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。

2.2　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。Mikiβ[3] 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。

3.1　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 の統語範疇は以下の表の通りである。

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

ただし、 `c` は $S, \bar{S}, NP, N$ のいずれかである。

X/Y という形式の統語範疇は `Fun1(X,Y,id)` と表し、その統語範疇を構成している統語範疇 X, Y のデー

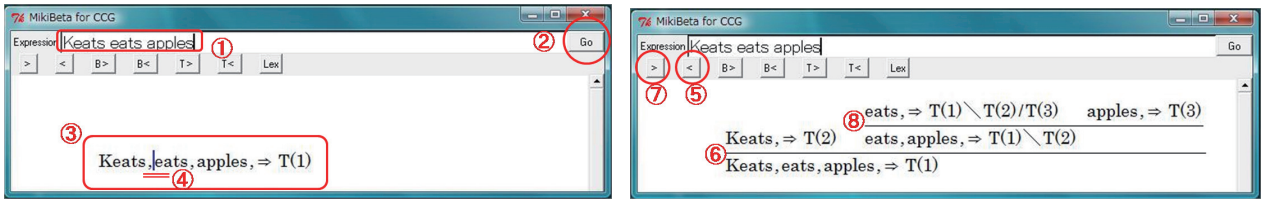

図 1　統語導出動作例

タを取得できるようにしている。id とは GUI 部がノードを識別するために定義されたものである。

3.2　動作例

実装の動作例を説明する (図 1)。

図 1 より、入力欄に解析する文を入力 (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、3.1 節で述べた T(n) で範疇を表す。

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、逆関数適用規則適用後の構成素が上段に表示される。“eats apples” についても同じように、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更新するようにする。

4　問題点と解決方法

4.1　型繰り上げ規則

2.1 節で述べた統語構造規則のなかで、以下の型繰り上げ規則 ($>T$,$<T$) 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。

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

例えば次のように、環境が “Keats eats” の場合は、

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

のように、“Keats” の型を $\boldsymbol{T}/(\boldsymbol{T}\backslash NP)$ として扱う。これに対して、型繰り上げ規則を証明木作成プログラムで使用した場合には以下のようになる。

$$>B\frac{>T\dfrac{Keats \Rightarrow T(4)}{Keats \Rightarrow T(1)/T(3)} \qquad eats \Rightarrow T(3)/T(2)}{\begin{array}{c}Keats, eats \Rightarrow T(1)/T(2)\\ \text{ただし、}T(3){=}T(1)\backslash T(4)\ \text{—(†)}\end{array}}$$

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ので、この「構造共有」の制約 (†) を保持しなければならない。

4.2　語彙項目と辞書

CCG では組み合わせ規則に加え、語彙項目も規則として扱われるため、規則が膨大な数となる。しかし先行研究 [3] では、各規則を適用する関数が規則ごとに定義されていたため、語彙項目を追加するためには膨大な数の関数を定義しなければならず、効率的ではない。

4.3　解決方法

そこで本研究では、規則そのものと規則を適用する過程を区別し、規則を単純なデータ構造で定義できるようにした。具体的には、語彙項目をデータとするクラス (Lexicon)、組み合わせ規則のデータ、規則を適用するクラス (Apply_rule)、4.1 の (†) のような制約が存在した時に、その制約を満たす最も一般的な統語範疇を計算するクラス (Unify)、Unify された情報をもとに証明木全体を書き換えるクラス (Rewrite_tree) などを定義し、どのような規則が与えられても適用することができる、より抽象度の高いアルゴリズムを書いた。

5　まとめと今後の課題

本研究では、[3] 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、新たなアルゴリズムを提案した。

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の導入も、今後の課題である。なお本研究は [4] において発表予定であり、詳しくはそちらを参照のこと。

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] 櫻井 加奈子, 浅井 健一.「汎用的に証明木を作成する『Mikiβ』」, 第 12 回プログラミングおよびプログラミング言語ワークショップ PPL2010(発表予定), 2010.

[4] 尾崎 有梨, 櫻井 加奈子, 浅井 健一, 戸次 大介.「証明木作成プログラムを用いた CCG 統語導出の実装」, 言語処理学会第 16 回年次大会発表論文集 (掲載予定), 東京大学, 2010.