自然言語処理 113－10
（1996．5．29）

高階関数に基づく範疇文法系の日本語への適用
― 表層と深層の語彙主導的な対応づけ ―

飯島 正, 関 洋平, 原田 賢一
(iijima@ae.keio.ac.jp)
慶應義塾大学 理工学部

筆者らは，範疇文法の新しいフォーマリズムとして，素性構造表現による高階関数に基づいた範疇文法系を既に提案している．そこでは，計算は，構文解析に相当する型推論と，意味解析に相当する意味関数の評価の二段階から構成される．本論文は，同範疇文法系を拡張して，格文法の考えを採り入れることにより，語彙情報として与えた表層格と深層格の対応関係を取り扱えるように拡張した結果を報告する．

An Application of a Higher-order-function-based Categorial Grammar System to Japanese

Tadashi Iijima, Yohei Seki, and Ken'ichi Harada,
Faculty of Science and Technology, Keio University

The authors have proposed a categorial grammar system based on higher-order functions which are specified in terms of feature structures. In the formalism the computation can be divided into two phases. The first phase, which corresponds to the syntactic analysis, is a type inference process. In the phase a semantic function for the given sentence may be composed. The second phase, which corresponds to the semantic analysis, is interpretation of the semantic functon. This paper describes an extension of the categorial grammar sytem. The extension incorporates the idea of Case Grammar into the system and enables lexical information to decide correspondence between surface cases and deep cases.

1 はじめに

本研究は，計算文法理論の一つである範疇文法 (Categorial Grammar)[1, 2] を取り上げ，その計算のためのメカニズム (TACG と略称する[1]) を構築することを目的としている．その基本的な考え方とプロトタイプの実行例は既に [11] で報告している．それは，範疇文法に対して，単一化文法の持つ語彙情報主導的な機構を採り入れて表現力の向上を図ることを目的としている．同じように，範疇文法と単一化文法の融合を図る研究に範疇単一化文法 (CUG; Categorial Unification Grammar)[5] があるが，TACG は，範疇文法の関数性を損なわないように，単一化文法で使われている素性構造を (高階的な) 意味関数の記述に採り入れている．本論文では，更に，その拡張として，格文法 (case grammar) の考え方を取り入れて，語彙に依存した格助詞の意味の選択を簡潔に記述することを試みた結果を報告する．

そのために行った計算モデル上の拡張点は，

- 素性構造において，属性ラベルとして変数を書けるようにする．
- $\vee$ 演算子を素性構造に与える (列挙した順序で優先順位も与える)．
- 意味マーカのチェックのための概念包摂関係 (subsumption) 述語 $\subseteq$ を与える．

[1] [11] では Typed Applicative Grammar と呼んできたが，略称を与えるにあたって TAG(Tree-Adjoining Grammar) と混同するので Typed Appricative Categorial Grammar(略称 TACG) と改称する

の三点である．

一方，言語データの側の拡張点は，基本的には，各動詞の辞書項目中の意味関数に，

- 表層格標識と深層格との対応付けと，
- 意味マーカ

を与えておく．表層格とは「が」格とか「を」格といった「音」に基づく格の表示のことをいい，表層格標識とは，「が」格とか「を」格に対する「が」や「を」のことである．また深層格とは，動作主格とか対象格といった意味に関する格の表示である．両者は一対一に対応するものではなく，動詞によってその対応の仕方が異なる．

こうした動詞の意味関数は，いわゆる格フレームに相当する．格フレームは一つの動詞に複数個与えられることがあり，その選択には意味マーカに基づく選択制限が使われることが多い．TACG でも，従来は，ある単語の範疇と意味関数を一対一に対応づけていたが[2]，「；」演算子によって優先順位付で複数の格フレームに対応する，複数の意味関数を与えることとする．

本報告では，まず，第 2 章で範疇文法と素性構造について，第 3 章で筆者らが既に提案している高階関数に基づく範疇文法系 TACG を概説した後，第 4 章で，本論文の提案である「語彙主導的な表層格と深層格の対応付け」のための拡張点について述べ，合わせて，日本語におけるいろいろな格助詞の取り方と，受身や使役といった態 (ヴォイス) に関する助動詞との関係についても触れる．

2 範疇文法と素性構造

2.1 範疇文法

範疇文法 [1, 2] は，自然演繹法 (natural deduction) やシーケント計算 (sequent calculus) といった論理計算における定理証明 (theorem proving) 過程を構文解析に対応付ける文法フォーマリズムである．そこでは，文が与えられたときに，それを構成する各語の語彙情報に含まれる統語範疇を抽出し，その統語範疇の列を仮定として推論規則 (inference rule) にしたがって書き換えていくことで構文解析が進められる．仮定である統語範疇の列から，文（の範疇）が構成できることが証明できたときに，構文解析が成功したことになる．

範疇文法では，文脈自由文法における文法規則のように語順を規定する要素は，統語範疇の表現の中に分解されて埋め込まれる．その範疇は，文法規則では終端記号に対応する各単語の辞書中の語彙項目の一つとして与えられる．範疇は，基本範疇と導出範疇に分けられる．ここでは，簡単な英語を想定して，基本範疇として，S(文)，NP(名詞句)，N(名詞) の三つを与える．すると，導出範疇は，/ (over と読む) と \ (under と読む) および括弧を使って基本範疇を組み合わせたものである．例えば，

NP/N	冠詞
$NP\backslash S$	自動詞
$(NP\backslash S)/NP$	他動詞
$(NP\backslash S)\backslash(NP\backslash S)$	副詞

となる．ここで，「値 / 引数」は「引数」を右側に与えると「値」を返す関数，「引数 \ 値」は「引数」左側に与えると「値」を返す関数の型とみなして理解することができる．この表記は Lambek によるものであり，\ の両辺を入れ替えた順序で表記する方法 (Steedman によるもの) も広く使われているので注意されたい (本報告では，筆者の好みにより Lambek による表記を採用している)．

上記のような範疇は，各語に対し，例えば，以下のように辞書中の語彙項目に範疇が与えられる．

単語	範疇
Hanako	NP
like	$(NP\backslash S)/NP$
Taro	NP
enthusiasticaly	$(NP\backslash S)\backslash(NP\backslash S)$

すると，文

$$\dfrac{\dfrac{\text{Hanako}\quad \text{likes}\quad \text{Taro}\quad \text{enthusiasticaly.}}{NP\quad (NP\backslash S)/NP\quad NP\quad (NP\backslash S)\backslash(NP\backslash S)}\qquad \dfrac{NP\backslash S}{NP\backslash S}}{S}$$

[2] 範疇が異なれば，異なる意味関数を与えていた

というようにボトムアップに書き換えていくことで文を構文的に解析できる.

その構文解析のための証明を行うための公理と推論規則である「Lambek のカリキュラス」を Gentzen 流のシーケント・カリキュラスの形式で示す [4]. 但し, 一般的な論理計算と異なり, 文を対象とするため, 仮定の順序が意味を持つ点に注意されたい.

$\frac{}{A \Rightarrow A}$	id 公理	$\frac{\Gamma \Rightarrow A \quad \Delta[A] \Rightarrow B}{\Delta[\Gamma] \Rightarrow B}$	カット
$\frac{\Gamma \Rightarrow A \quad \Delta \Rightarrow A \backslash B}{\Gamma, \Delta \Rightarrow B}$	$\backslash$ 右 (除去)	$\frac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A \backslash B}$	$\backslash$ 左 (導入)
$\frac{\Delta \Rightarrow B / A \quad \Gamma \Rightarrow A}{\Delta, \Gamma \Rightarrow B}$	/ 右 (除去)	$\frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow B / A}$	/ 左 (導入)
$\frac{\Gamma \Rightarrow A \cdot B \quad \Delta[(A, B)] \Rightarrow D}{\Delta[\Gamma] \Rightarrow D}$	$\cdot$ 右 (除去)	$\frac{\Gamma \Rightarrow A \quad \Delta \Rightarrow B}{\Gamma, \Delta \Rightarrow A \cdot B}$	$\cdot$ 左 (導入)

本報告では, id 公理とカット, およびドット対 (·) を省き, / と \ の除去ならびに導入 (の一部) を取り上げる. (id 公理はイニシャル・シーケントの導入に使われている. また, 上記カリキュラスにおけるカット除去に対する取り組みは既に Lambek に与えられている).

2.2 素性構造の表記と操作

ここでは, 素性構造の表記と操作を簡潔に紹介する. 素性構造は, 「属性ラベル＋属性値」対の集合であり. 以下のように記述する.

$$\begin{bmatrix} \text{ラベル}_1 : & \text{値}_1 \\ \text{ラベル}_2 : & \text{値}_2 \\ \vdots & \vdots \\ \text{ラベル}_n : & \text{値}_n \end{bmatrix}$$

属性の出現順序には意味がなく, 入れ子にする (属性値として素性構造をとる) ことが可能である. また, 素性構造は, いわゆる部分指定項 (partialy specified term) であり, 指定されていない部分は値が未定であるか, 単にその場で指定する必要がないことを意味しているに過ぎない. ここでは素性構造に関する演算として, 単一化操作だけを設ける. 単一化操作は, 二つの素性構造が等価となるように, 値が未定の属性に対し代入を行う操作である. 入れ子構造があるときには, その構造にしたがって再帰的に双方向の代入が行われる. ある属性に対する属性値が互いに異なる場合には, 値が未定の属性に代入を行うだけでは互いに同じ値とすることができないので, 単一化は失敗する.

3 高階関数に基づく範疇文法系 TACG の計算

高階関数に基づく範疇文法系 TACG の計算は次の 2 ステップで行われる [11]:

- ステップ 1 (構文解析) →型推論による意味関数の導入
- ステップ 2 (意味解析) →意味関数の計算

ここで, ステップ 1 の計算は, 厳密には一般的に言う型推論とは異なる. 一般に型推論とは, 関数の適用と定義の構造が与えられたとき, 型の整合性をチェックするものである. しかし, このステップ 1 の計算では, 適用構造から線形的な順序へ縮退している「文」と, 一つの語が複数の範疇を持つことがある多相的 (polymorphic) な語彙情報である「辞書項目」を与えられて, そこから整合的な関数の適用構造, すなわち「構文木」を再生する計算である. しかし, 範疇と意味関数の型とみなすことで, 型の推論を行うことに他ならないということから, 「型推論」と呼ぶこととする. こうした関数プログラムと証明との対応付けは Curry-Howard の同型対応として知られているが, ここでは, 更に, 推論の結果得られた証明図は, 与えられた文の構文構造にも対応している.

3.1 型推論の実行メカニズム

TACG のプロトタイプでは, **2.1** 節でシーケント・カリキュラス風の表記で示した Lambek のカリキュラスの中で, id 公理とカット, およびドット対 (·) を省き, / と \ の除去を取り上げて, Prolog 上に実装した (id 公理はイニシャル・シーケントの導入に使われている. また, 上記カリキュラスにおけるカット除去に対する取り組みは既に Lambek に与えられている). また, 一つの単語が複数の範疇を持ち得るという多相性 (polymorphism) を扱うために, オペレータ ";"(∨ として知られている [4]) とその除去規則を与えた. 紙数の都合上,

詳細は省くが，英語なら “with” が名詞にかかる場合には範疇 $(n\backslash n)/np$, 動詞にかかる場合には
範疇 $((np\backslash S)\backslash(np\backslash S))/np$ を持つので，その両者を “;” でつないだ範疇を与えておかねばならない（除去規則は一方を切り捨てる働きをする）．

3.1.1 / と \ の除去規則

除去規則の実装は Prolog 上では非常に容易であり，以下のどちらかの上式のパターンを見つけたら下式で置き換えるだけである．もちろん，バックトラックによって，複数の置換候補は順次，試される．

$$\frac{\cdots\ A\ \ A\backslash B\ \cdots}{\cdots\ B\ \cdots} \qquad \frac{\cdots\ B/A\ \ A\ \cdots}{\cdots\ B\ \cdots}$$

実際には，同時に意味関数の導入を行う．範疇 A の意味関数を g, 範疇 B の意味関数を f として，範疇の後ろに “:” で区切って意味関数を付加し，引数 x への関数 f の適用を $(f\ x)$ と表記するとするなら，以下のように意味関数の導入を行う．

$$\frac{\cdots\ A{:}g\ \ A\backslash B{:}f\ \cdots}{\cdots\ B{:}(f\ g)\ \cdots} \qquad \frac{\cdots\ B/A{:}f\ \ A{:}g\ \cdots}{\cdots\ B{:}(f\ g)\ \cdots}$$

除去規則によって下式中の範疇の個数は，上式よりも必ず減少するので，この置換は，いずれ停止する．

3.2 意味関数の表記法と実行

TACG では，意味関数を，引数に素性構造を一つとり，値として素性構造を一つ返すような関数で与える．更に，その関数定義自体を素性構造として表記する．意味関数は一般に高階関数となるが，そうすることで，簡潔に高階関数を定義することができる．本報告では，プロトタイプの実行に必要最小限の構成要素しか示さない．

3.2.1 意味関数の表記法 (「関数の型」としての範疇)

意味関数は，型として統語範疇をとる．そこで，範疇の表現を，以下のように関数の型と捉えるものとする．

範疇		型
基本範疇	A	基本型
導出範疇	A/B	$A \leftarrow B$
	$B\backslash A$	$A \leftarrow B$

但し，「$\leftarrow$」の右辺を「引数の型（定義域）」，左辺を「値の型（値域）」とする．基本型は，関数に対する「定数関数」（すなわち，「引数をとらない関数」）を意味する．

ここで，上記の表における範疇 A, B は型変数であり，値として導出範疇をとれることに注意されたい．例えば，$(A/B)\backslash C$ というような範疇は許されて，$C \leftarrow (A \leftarrow B)$ というような高階関数を意味する．

3.2.2 意味関数の表記法 (関数の定義法)

関数の定義（λ 式に対応）は，以下のような構文で素性構造として与える．

```
素性構造   ::=  関数定義 | 定数関数 | 項.
関数定義   ::=  [ var :  素性構造 ]
                [ val :  素性構造 ] .
定数関数   ::=  [ ラベル_1 :  素性構造
                  ラベル_2 :  素性構造
                      :           :
                  ラベル_n :  素性構造 ] .
```

但し，ラベル$_i$ は，var と val 以外とし，互いに異なるものとする．また，「項」は定義していないが，Prolog における任意の項構造（上記の素性構造と混同しないもの．変数も含む）を想定してほしい．

ここで，var と val は，それぞれ「関数の仮引数」と「関数の値」を意味する．本報告では，プロトタイプに必要最小限の要素だけを示しているため，条件分岐などのコンストラクトは一切与えていない．

仮引数が素性構造の形をしているのは，次のセマンティックスの項で詳述するが，関数適用の際に実引数との単一化によって代入を行うためである．引数や関数値として「関数定義」をとれることから，高階関数を取り扱うことができる点に注意されたい．

3.2.3 意味関数の表記法 (関数適用のセマンティックス)

この項では，意味関数のインタープリタの主な動作について説明する．以下では，解りやすさのために，関数定義 $\begin{bmatrix} \text{var}: & X \\ \text{val}: & Y \end{bmatrix}$ を $\lambda X.Y$ と表記するものとする．関数適用 $(\lambda X.Y\ Z)$ は，以下のように行う（但し，X, Y, Z は素性構造とする）．

- 仮引数 X と実引数 Z を単一化する．この際，X の属性値に Y の属性値と共有している変数があれば，そこに単一化代入の効果が伝播される．
- その単一化に失敗したら，関数の値を $\perp$ とする．
- 単一化に成功したら，Y を値として返す．
- いかなる素性構造も $\perp$ との単一化は常に失敗する．

3.3 従来の TACG における日本語の解析例

ここで示す例は，日本語特有の語順の柔軟性に対する取り扱いを提示することを目的としており，単純な例文しか取り扱うことのできない語彙情報・文法情報である．
入力文を例えば，以下のように与えるものとする．

```
[花子，を，太郎，が，愛す，ている，た]
```

辞書の項目 (語彙情報) は以下のように考える．

```
dic(花子,     n,     [意味:花子] ).
dic(太郎,     n,     [意味:太郎] ).
dic(愛す,     v,     [意味:愛す]  ).
dic(が,     n\(s/s),
        [var:[意味:X],
         val:[var:[意味:M,時制:T,様相:A,対象格:P,時格:Q,場所:R],
              val:[意味:M,時制:T,様相:A,主題格:X,動作主格:X,対象格:P,時格:Q, 場所:R]]] ).
dic(を,     n\(s/s),
        [var:[意味:X],
         val:[var:[意味:M,時制:T,様相:A,動作主格:O,時格:Q,場所:R],
              val:[意味:M,時制:T,様相:A,主題格:X,動作主格:O,対象格:X,時格:Q, 場所:R]]] ).
dic(た,     v\(s;v),
        [var:[意味:X,様相:A], val:[意味:X,様相:A,時制:過去]] ).
dic(ている,     v\(s;v),
        [var:[意味:X, 時制:T], val:[意味:X,様相:継続,時制:T]] ).
```

これに対して与えられる意味関数は，語彙項目 X の意味関数を $sem[X](\text{引数})$ で表したならば，

$$
\begin{array}{l}
(sem[\text{を}](sem[\text{花子}]()) \\
\quad (sem[\text{が}](sem[\text{太郎}]()) \\
\quad\quad (sem[\text{た}] \\
\quad\quad\quad (sem[\text{ている}](sem[\text{愛す}]()))))))
\end{array}
$$

となる．これを計算すると，実行結果は，

[意味: 愛す, 時制: 過去, 様相: 継続, 主題格:_324, 動作主格: 太郎, 対象格: 花子, 時格:_386, 場所:_394]

というように得られる．ここで，「を」格と「が」格の語順によらず，意味関数中の格スロットに値が与えられている点に注意されたい．日本語の語順に関して，このアプローチでは，意味関数を格フレームとして，構文解析しながら格標識（格助詞）に基づいて，それを構築していく形で柔軟に取り扱うことができる．

4 語彙主導的な表層格と深層格の対応付け

4.1 語彙情報

前章で与えた解析例では，名詞や動詞[3]に関する語彙情報としては，「意味」属性しか与えていない．範疇を見ても，名詞 n と動詞 (用言) v は引数をとらない関数，すなわち定数 (関数) であって従属的な役割しか果たしていないように見える．それに対して，格助詞 $n\backslash(s/s)$ と助動詞 $v\backslash(s;v)$ は，各々，名詞と動詞 (用言) を引数にとる関数として主辞的な役割を果たす．格助詞は，名詞を引数に取って，格後置詞句 s/s を返す．一方，助動詞は，動詞 (用言) を引数に取って，文 s または動詞 v を意味する $s;v$ を返す．その返却値は，実質的には動詞句 (用言句) に相当するが，格の省略を許容するために文 s として扱っている．この様子を図示すると図1のようになる．ここで，動詞句 (用言句) に相当する「文」が，関数として引数である「格後置詞句」をとるの

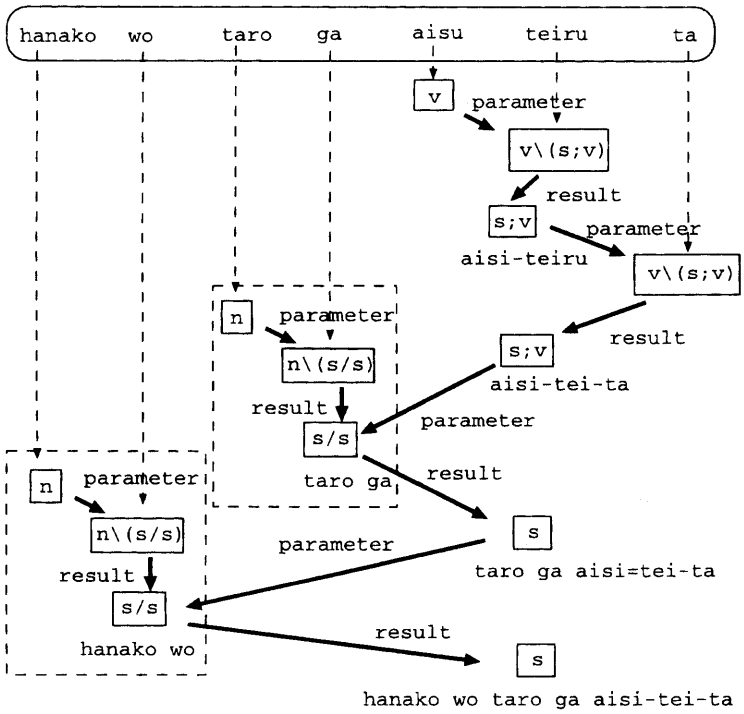

図 1: TACG における計算の様子

ではない点に注意して欲しい．「格後置詞句」の方が関数として，引数「文」に適用されて，値として新たに「文」を生成する．その際に，「格後置詞句」は，引数の「文」に自分自身に関する情報を追加して，新しい「文」を生成する．これによって，「格の出現順序」が自由になるだけでなく，「格の省略」があっても対応することができる．

[3]実際には，形容詞，形容動詞を含めた用言

4.2 従来の方法の問題点

しかし，ここに幾つかの問題がある．一般に，格助詞の持つ意味は動詞に依存して決まる．たとえば，表層格「に」格は，動詞「行く」に対しては「行く目的地」を意味し，動詞「会う」に対しては「会う相手」を意味する．すなわち，フィルモアの深層格でいえば「目標格」と「対象格」となる[4]．また，所有 (東京にはビルが多い)，可能 (彼には英語が分かる)，必要 (釣には餌が必要だ) といった状態述語文のように，「に (には)」格が主格を与え，「が」格が対象格を与えるような，いわゆる与格主語構文も知られている [6]．こうした表層格と深層格の対応の語彙依存性を扱うためには，動詞 (用言) の語彙情報として「意味属性」だけでなく，その対応関係も保持しておき，「格後置詞句」関数の適用時にチェックできなければならない．

更に，同じ表層格が異なる深層格に対応して複数回出現するケースもある．道具格と場所格の「で」が二重に出現するケース (デパート で クレジットカード で 洋服を買った) や，時間格と対象格の「に」が二重に出現するケース (3時 に 彼 に 会う)，特殊なケースで場所と対象の「を」が二重に出現するケース (雨の街 を 迷い猫 を 探して歩いた) もある．こうした場合には，意味マーカの利用が役立つ．時間や場所は，あまり動詞に依存することはないと判断できるが，道具や対象は動詞 (用言) に依って大きく異なる．しかし，「太郎 が 花子 が 好きだ．」のように，意味マーカでも判断が付かず，用言からの距離に基づいて判断するしかないものもある．それでも，埋め込み文と組み合わされると，「太郎 が 花子 が 好きだと言った．」のように「『花子 が 好きだ』と太郎 が 言った．」ケースと「『太郎 が 花子 を 好きだ』と言った．」というケースが判断できないこともある．

4.3 語彙主導性を導入する方法

基本的には，用言の語彙情報として，表層格の標識を属性とし，深層格と意味マーカの組を属性値として与えておく．たとえば，

```
dic(愛す, v, [意味: 愛す, が格: 動作主格 (人), を格: 対象格 (オブジェクト)]).
```

としたとき，表層格「を」格に対応する深層格は「対象格」で，意味マーカは任意のオブジェクトということを意味する．それに対して，格助詞「を」の語彙情報を以下のようにすることで，

```
dic(を,     n\(s/s),
        [var:[意味:X],
         val:[var:[意味:M, 時制:T, 様相:A, が格:G(SemG), を格:W(X)...],
              val:[意味:M, 時制:T, 様相:A, が格:G(SemG), W:X... ]]] ).
```

引数に取る名詞を意味マーカと単一化 (ここは実際には，概念の包摂関係 (subsumption) に基づく単一化にすべきである．が，紙数の都合上，詳細は略す)．し，なおかつ，「を」格に対応する深層格を導入して属性値を設定している．

更に，素性構造に $\vee$ を導入し，それに対応する単一化の拡張を行う．紙数の都合上，直観的な表現に留める．単一化を $\cup$ で表すとすると，

$$
\begin{aligned}
(X_i \vee Y) \cup (X_1 \vee \cdots \vee X_i \vee \cdots X_n) &\rightarrow Y = (X_1 \vee \cdots \vee X_(i-1) \vee X_{i+1} \vee \cdots X_n) \\
(X_i \vee Y) \cup X_i &\rightarrow Y = \bot \\
(X_i \vee Y) \cup X_j &\rightarrow \text{失敗 (但し，} X_i \neq X_j)
\end{aligned}
$$

というように，優先順位の順序を変えずに，前方から一致した要素を抜き出すように働く．これによって，

```
dic(会う, v, [意味: 会う, に格:(対象格 (人) ∨ 時間格 (時刻)),...]).
dic(に,     n\(s/s),
        [var:[意味:X],
         val:[var:[意味:M, 時制:T, 様相:A, に格:(N(X) ∨ NN)...],
              val:[意味:M, 時制:T, 様相:A, に格:NN, N:X... ]]] ).
```

というように複数の表層格を与えて意味マーカにマッチしたものを選択することができるようになる．

[4] 但し，深層格としてどういうものを選ぶべきかについては諸説あり，また格の選択もゆらぎがあって一意に選択できないともいわれている [7].

4.4 検討

上記のように表層格と深層格を対応づけることで，「彼の書いた本」といった連体修飾における「の」を動作主格と結びつけることが可能になったり (「彼の沈没した船」では「の」は主格にならないことに注意)，受身文や使役文などのように助動詞によって格の役割を変換させる [12, 8] ような場合に，深層格を使うことでより的確な処理ができるようになった．

5 おわりに

筆者らの提案による高階関数に基づく範疇文法系の拡張として，語彙情報主導的な深層格選択を簡潔に記述する方法について述べた．このアプローチには，まだまだ採り入れるべき言語学上の成果が多く残されている．そればかりか，属格の「の」を含めて，副詞，接続詞など，まだ取り扱っていない構文要素が数多く残されている．それらを，順次取り上げていくつもりである．筆者らのグループでは，範疇文法をベースに新しい計算のモデルを構築することを試みる本プロジェクトと並行して，既存の CUG(範疇単一化文法) の枠組をベースに日本語への適用と実用化を考慮したプロジェクトを進めている [12]．今後も，両プロジェクトで得られた知見を相互に交換しつつ作業を進める予定である．

また，範疇文法の構文の枠組の上に内包論理に基づいた意味表現を展開したものとしてモンタギュー文法がある [10][5]が，今後，筆者らも意味表現に関しても検討を加えて行きたい．

最後に，HPSG(Head-driven Phrase Structure Grammar)[9] との関係を簡単に示す．本研究での意味関数の評価において，制約をチェックしながら素性構造に情報を収集していくという枠組は，HPSG から多くの影響を受けている．特に，語順の自由さを与える点において顕著である．しかし，範疇文法の枠組の上に，高階関数によって関数適用で関数定義 (定数を含む) を書き換えていく本研究のアプローチは独自のものである．特に，助詞や助動詞が関数として，名詞や動詞 (用言) の定義を書き換えていく点は，HPSG における主辞が補語の情報を吸い上げていく様子と，興味深い対比をなしている．

参考文献

[1] Mary McGee Wood: *"Categorial Grammars,"* Routledge (1993).

[2] W. Buszkowski, W. Marciszewski, and Johan van Benthem : *"Categorial Grammar,"* Linguistic and Literary Studies in Eastern Europe Vol.25, John Benjamins Pub.(1988).

[3] Joachim Lambek: *"The mathematics of sentence structure,"* American Mathematical Monthly 65(1958), also in [2].

[4] Glyn V. Morrill: *"Type Logical Grammar - Categorial Logic of Signs,"* Kluwer Academic Pub.(1994).

[5] Hans Uszkoreit: *"Categorial Unification Grammars,"* Proc. of COLING'86,pp.187-194(1986).

[6] 三原 健一: "日本語の統語構造 — 生成文法理論とその応用," 松柏社 (1994).

[7] 仁田 義雄: "日本語の格をめぐって," くろしお出版 (1993).

[8] 長尾 真: "画像と言語の認識工学," コロナ社 (1989).

[9] カール・ポラード，アイバン・A・サグ (郡司 隆男 訳): *"HPSG* 入門 — 制約にもとづく統語論と意味論," 産業図書 (1994).

[10] 松本 裕治，田中 穂積: "日本語モンテギュ文法の実働化と質問応答への応用," 情報処理学会 自然言語処理研究会, Vol.31-7(1982).

[11] 飯島 正，関 洋平，柳原 正秀，木下 知貴，原田 賢一: "範疇文法のための関数的な計算機構," 情報処理学会 自然言語処理研究会, Vol.111-6, pp.33-40(1996).

[12] 関 洋平，飯島 正，原田 賢一: "範疇単一化文法を日本語に適用する際の付加的な規則について," 情報処理学会 自然言語処理研究会, Vol.113(1996)(本号収録予定).

[5] 文献を御紹介くださった NTT の小暮潔氏に深く感謝いたします．