自然言語処理 111－6
（1996. 1. 19）

範疇文法のための関数的な計算機構

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

本論文は，計算文法理論の一つである範疇文法を取り上げ，その計算のためのメカニズムを構築することを目的としている．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．型推論において導入する意味関数は一般に高階関数となるが，その定義の表記法を素性構造とすることで，簡潔に表現することができる．簡単な英語と日本語の解析例を示している．

A Typed Functional Computation Mechanism for the Categorial Grammar Formalism

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

This paper describes a computation mechanism for the categorial grammar. 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. The semantic function may be constructed by higher-order functions, which can be specified in terms of feature structures. This paper also includes some examples to analyse simple English and Japanese sentences.

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（Typed Applicative Grammarと呼んでいる）を構築することを目的としている．

範疇文法は，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ものである．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．

このとき，範疇としては，原子的な記号（基本範疇）だけでなく，(型付き)関数定義に似た表記をつかって基本範疇から組み合わせてつくる導出範疇の表記法を与える．それによって構造的な表現を与えることが可能となり，範疇文法では，ボトムアップに範疇の列の項書換えを行っていくことで，文の構文的な解析ができる．更に，意味解析を考慮するとき，この構文解析は，意味を計算する関数のための型推論とみなすことができる（厳密には，一般的な型推論とは異なっている．その点については後述する）．ここで，範疇は関数の型とみなされる．

一方，同様に定理証明過程を構文解析に対応付けるアプローチに限定節文法(DCG)[6]がある．しかし，範疇文法は語彙と対応付けた統語範疇の書き換えを繰り返す語彙情報主導の解析を行うのに対し，限定節文法は文脈自由文法の文法規則をProlog（論理プログラミング）のプログラムの構成要素であるホーン節と対応づけて，(トップダウンに）規則の適用を繰り返す文法規則主導の計算を行う．ここでも，構文解析は，構文規則の項書換えとして行うことができる．

更に，意味解析を考慮すると，限定節文法には，単一化文法(Unification Grammar)[5]と呼ばれる興味深い拡張が知られている．単一化文法は，構文規則を論理式として表現する限定節文法の枠組みに，単一化操作を定義したレコード表現（「属性名ラベル＋属性値」対のリスト）である素性構造（feature structure）を取り入れて語彙情報（lexicon）の表現力を拡張したものである．その素性構造をつかって各単語のもつ性質を，同じ構文規則に現れる他の語の性質と制約として結び付けることにより，制約に基づく意味解析（の一部）を構文解析と同時に行うことができる．

範疇文法と限定節文法では，関数プログラミング的なセマンティックスと論理プログラミング的なセマンティックスという違い，そして，語彙主導のボトムアップ計算と文法規則主導のトップダウン計算という違いこそあれ，両者とも項書換えに基づく計算をベースにしている点は一致している．そこで，むしろ文法規則主導と語彙主導という相補性を積極的に取り入れるという意味で，限定節文法に語彙情報の表現力の拡充を図ったものとして，単一化文法を位置付けることができる．

更に，この両者の利点を同時に採り入れるために，両者を融合する研究アプローチの一つにCUG (Categorial Unification Grammar) [7] がある．CUGにおける両者の融合は，単一化文法の枠組みの中に範疇文法をコーディングして埋め込むことで行われている．しかし，このアプローチでは，範疇文法の持つ関数性が失われてしまい，情報の流れの方向の明解さが失われてしまう．

そこで本報告では，範疇文法と単一化文法を融合するという共通の目的のもとで，むしろ逆に範疇文法の枠組の中に素性構造とその単一化操作を埋め込むことで更に語彙情報の表現力の強化を図ることを試みる．本報告で提案する計算機構では，計算は次の２ステップで行われる：

- 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．

本論文では，意味関数（一般に高階関数となる）の表記法を定義し，このステップ1の推論プログラムとステップ2の意味計算のインタープリタの基本部分（プロトタイプ）をPrologで記述した結果を報告する．本研究プロジェクトの全体的な構想では，このステップ1とステップ2を，プログラム等価変換技法により，融合することで，構文解析と意味解析を統合することを計画しているが，本報告には，それは含まない．

以下では，２章で範疇文法を概説し，３章で単一化文法の紹介とCUGにおける両文法記述の融合方法について解説した後，４章で型推論の実行メカニズムについて，５章で意味関数の表記法とインタープリタの実行メカニズムについて，述べる．

2 範疇文法

2.1 範疇文法とは

範疇文法[1, ?]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

$$\begin{array}{cccc} \text{Hanako} & \text{likes} & \text{Taro} & \text{enthusiasticaly.} \\ NP & (NP\backslash S)/NP & NP & (NP\backslash S)\backslash(NP\backslash S) \\ & \multicolumn{2}{c}{\overline{\qquad NP\backslash S \qquad}} & \\ & \multicolumn{3}{c}{\overline{\qquad\qquad NP\backslash S \qquad\qquad}} \\ & \multicolumn{3}{c}{\overline{\qquad\qquad\qquad S \qquad\qquad\qquad}} \end{array}$$

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．

2.2 Lambekのカリキュラス

Lambekのカリキュラスは，以下のような規則として与えられている[3]．

適用	$X/Y\ Y \Rightarrow X$ $Y\ Y\backslash X \Rightarrow X$	上昇	$X \Rightarrow Y/(X\backslash Y)$ $X \Rightarrow (Y/X)\backslash Y$
結合則	$(X\backslash Y)/Z \Rightarrow X\backslash(Y/Z)$	商	$X/Y \Rightarrow (X/Z)/(Y/Z)$ $X\backslash Y \Rightarrow (Z\backslash X)\backslash(X\backslash Z)$
合成	$X/Y\ Y/Z \Rightarrow X/Z$ $Z\backslash Y\ Y\backslash X \Rightarrow Z\backslash X$	逆商	$X/Y \Rightarrow (Z/X)\backslash(Z/Y)$ $X\backslash Y \Rightarrow (Y\backslash Z)/(X\backslash Z)$

ここでは，Curry-Howardの同型対応を強調する意味で，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．

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

3 単一化文法とCUG

3.1 素性構造の表記と操作

ここでは，単一化文法についての解説は省き，素性構造の表記と操作だけを紹介する．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．

単一化文法では，文法規則に現れる非終端記号や終端記号(および，それに対応する各単語の辞書中の語彙項目)に，こうした素性構造を用いることで，単語の字面表現，意味項目，音韻表現などに加えて，文法的な制約(性・数の一致など)の検査に必要な属性を取り扱うことができる．

3.2 範疇単一化文法(CUG)

1章で述べたようにCUG[7]では，導出範疇を素性構造へマッピングすることで範疇文法を単一化文法に取り入れている．例えば，他動詞に対応する範疇は，

$$(NP\backslash S)/NP$$

というように与えられるが，それは以下のような素性構造となる(当初の論文[7]ではPATR形式のdag(directed acyclic graph)表現で与えられている)．

$$\begin{bmatrix} \text{結果:} & \begin{bmatrix} \text{結果:} & S \\ \text{方向:} & \backslash \\ \text{引数:} & NP \end{bmatrix} \\ \text{方向:} & / \\ \text{引数:} & NP \end{bmatrix}$$

もちろん，本来は他にも多くの素性が含まれており，性・数の一致といった制約を伝播するためにも使われる．

4 型推論の実行メカニズム

今回報告するプロトタイプでは，**2.2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 $(\cdot)$ 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)．/と\の導入規則については，検討だけを与える．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）．

4.1 /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．

4.2 /と\の導入規則

2.2節で与えた導入規則は，自然演繹流に表記すると，以下のような仮説推論になることに注意して欲しい．

$\dfrac{\begin{array}{c}[A],\Gamma\\ \vdots\\ B\end{array}}{A\backslash B}$	\導入	$\dfrac{\begin{array}{c}\Gamma,[A]\\ \vdots\\ B\end{array}}{B/A}$	/導入

すなわち，前か後にAの存在を仮定してBが導出できたとき，$A\backslash B$かB/Aを導く推論規則が，これら「導入規則」（シーケント・カリキュラス風に呼べば「右規則」）である．

しかし，このような導入規則を安易に実装すると，無意味な仮説を次々に導入して推論が停止しなくなってしまう．そこで，範疇の置換が隣接した範疇との相互作用でしか発生しないことを利用して，導入規則が必要となるパターンを分類してみる．すると，導入規則が繰り返されず導入の次に除去が行われるとすれば，以下のようなパターンは見出すことができる（もちろん，これで十分ではない）．隣接している要素からX,Yは決まるので，$\cdots\ A\ Y$や$Y\ A\cdots$からXが導出できるかどうかを判定すればよく，それ自身は型推論手続きを再帰的に適用すればよい．

$\dfrac{\dfrac{\cdots\ A\ [Y]\ \ (X/Y)\backslash Z}{\begin{array}{c}\vdots\\ X\end{array}}}{X/Y}$	$\dfrac{\dfrac{\cdots\ [Y]\ A\ \ (Y\backslash X)\backslash Z}{\begin{array}{c}\vdots\\ X\end{array}}}{Y\backslash X}$
$\dfrac{Z/(X/Y)\ \ \dfrac{A\ [Y]\ \cdots}{\begin{array}{c}\vdots\\ X\end{array}}}{X/Y}$	$\dfrac{Z/(Y\backslash X)\ \ \dfrac{[Y]\ A\ \cdots}{\begin{array}{c}\vdots\\ X\end{array}}}{Y\backslash X}$

5 意味関数の表記法と実行

前章で示した素性構造自身は，ラベル名称に「引数」とか「結果」とあるにもかかわらず,「計算」とは独立した中立的な表現といえる．しかし，この素性構造を関数の定義とみなして直接実行するような計算セマンティックスを与えることは可能である．この章では，そのための意味関数の表記法（構文と意味）の一部を直感的に与える．

本研究では，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．また，個々の意味関数の内容は，対象とする言語の構文（範疇の設定）と意味表現に依存して決まるので，本章では英語と日本語を例にとって，簡単な例を与える．

5.1 意味関数の表記法の概略

5.1.1 「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．

5.1.2 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	::=	$\left[\begin{array}{ll} \text{var}: & \text{素性構造} \\ \text{val}: & \text{素性構造} \end{array}\right]$.
定数関数	::=	$\left[\begin{array}{ll} \text{ラベル}_1: & \text{素性構造} \\ \text{ラベル}_2: & \text{素性構造} \\ \vdots & \vdots \\ \text{ラベル}_n: & \text{素性構造} \end{array}\right]$.

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．

5.1.3 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$\left[\begin{array}{ll} \text{var}: & X \\ \text{val}: & Y \end{array}\right]$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．

5.2 英語の例

本項では**2章**で与えた例

"Hanako likes Taro."

を再度取り上げる．辞書中の語彙情報に以下のように意味関数の定義を与えるものとする．

```
dic(hanako,  np,      [人称:3, 数:単, 意味:花子]).
dic(taro,    np,      [人称:3, 数:単, 意味:太郎]).
dic(loves,   (np\s)/np,
        [var:[意味:Y],
         val:[var:[人称:3,数:単,意味:X],
              val:[時制:現在,形態:三単,意味:愛す(X,Y)]]]).
dic(enthusiasticaly,  (np\s)\(np\s),
        [var:[var:[人称:P,数:N,意味:X],
              val:[時制:T,形態:F,意味:M]],
         val:[var:[人称:P,数:N,意味:X],
              val:[時制:T,形態:F,意味:強く(M)]]]).
```

2章で示したように型推論すると同時に，この入力文に対応して，次のような意味関数が生成される(Prologで試作したプロトタイプの出力に読みやすいように整形した).

```
(
    ([var:[var:[人称:_648,数:_656,意味:_664],
           val:[時制:_698,形態:_706,意味:_714]],
      val:[var:[人称:_648,数:_656,意味:_664],
           val:[時制:_698,形態:_706,意味:強く(_714)]]] ,

     ([var:[意味:_336],
       val:[var:[人称:3,数:単,意味:_374],
            val:[時制:現在,形態:三単,意味:愛す(_374,_336)]]] ,
      [人称:3,数:単,意味:太郎])),

   [人称:3,数:単,意味:花子])
```

これを評価すると以下のような結果が得られる.

```
[時制:現在,形態:三単,意味:強く(愛す(花子,太郎))]
```

5.3 日本語の例

筆者らのグループでは，本プロジェクトと並行した別のサブ・プロジェクトとして，範疇文法と日本語の相性を調べることと日本語の諸現象の収集・整理を兼ねて，CUG形式をベースとした日本語のための範疇単一化文法を実験的に構築している．その結果は，別途，報告する予定である[8]が，本章で与える日本語文法はそれとは異なり，本プロジェクトのアプローチ(Typed Applicative Grammarと呼んでいる)のデモンストレーションを目的として，このアプローチに適した形の文法を与えたものである．そのため活用語尾を取り扱っていないなどといった点では，一般性に欠ける面があることは予め申し述べておく．この項では，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いる.

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]] ).
```

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，ヲ格とガ格の語順によらず，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．

6 おわりに

本報告のアプローチ自体は極めて素朴であり従来なされてきた研究の自然な延長上にあるが(範疇文法自体の有用性が必ずしも明確ではないというのは別にしても)，実際にはまだ多くの課題が残されている．たとえば，関数定義のために素性構造を使うためには，素性構造にどのようなコンストラクトを追加すべきであるかも明確ではない．少なくとも，帰納的な定義のためには，選択関数(disjunction)が必要である．また，各種の文法的制約を評価するのに計算順序は，ボトムアップ（最内優先）だけで十分なのだろうか? 少なくとも，範疇文法において興味深い演算の一つであるタイプレージング[1]等を直接的に扱うためには代数的等価性にしたがった変換を取り入れる必要がある．こうした問題は，一般的な計算能力の解析とは別に，実際に文法記述を試みながら，ニーズ指向で解消していくことが必要であるとも考えられる．

また，今後の課題として，今回，試作している型推論と意味関数のインタープリタをプログラム変換手法を用いて融合することを試みたいと考えている．

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] Sturt M. Shieber: *"An Introduction to Unification-Based Approaches to Grammar,"* CSLI Lecture Notes No.4(1986).

[6] F.C.N. Pereira and D.H.D. Warren: *"Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Transition Networks,"* Artificial Intelligence, Vol.13, pp.231-278(1980).

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

[8] 関 洋平，飯島 正，原田 賢一: "範疇単一化文法の日本語への適用," 情報処理学会第52回全国大会(1996)（発表予定）.