4. 自然言語処理

東京工業大学　田中穂積

1・ はじめに

コンピュータに自然言語を理解させる自然言語理解システムの研究が近年盛んになってきている。これは、人間とコンピュータとの間の使用言語の相違に基づくセマンティック・ギャップを解消して、人間とコンピュータとの間に新しいインタフェースを確立しようとするものである。第5世代コンピュータ計画では、自然言語理解システムは、知的インタフェースの研究の一部として含まれている。自然言語処理はこの研究に密接に関連している。言語を理解するためには言語の深い処理が必要になる。広範な言語現象を深く処理する技術はまだ確立されてはいない。そのため現状では、データベースをアクセスするための質問文などに限定し、扱う言語現象を狭く限定して、その範囲で質問文の深い処理を行なわざるを得ない。

機械翻訳システムの研究が最近我が国で盛んに行なわれている。これは、自然言語処理の総合的な応用システムの一つである。高度な機械翻訳システムを実現するためには、自然言語理解システムと同様に、翻訳する言語の深い処理が必要なことは明らかである。しかし機械翻訳システムは、自然言語理解システムと異なり広範囲な言語現象を扱わなければならないため、浅いレベルの処理にとどめざるを得ないというのが現状である。一方自然言語の文を生成する研究が良く行なわれている。

以下では自然言語処理に関連する諸技術を構文解析と意味解析に焦点をあてて順に説明する。文脈解析については多くの研究課題が残されている。

2・ 自然言語処理を巡る諸知識

自然言語処理を行なうための知識を大きく分けると次の2つがある。

［1］言語学的な知識（辞書に記載された知識や文　　法）

［2］それ以外の知識（常識等）。

［1］については言語学者が様々な角度から研究している。とりわけ文法については、言語学者が詳しく研究しており、言語学的な理として多くの知見が得られている。文法の言語学者によって明らかにされた文法規則にする知識については、それらを部分的に利用た自然言語処理システムも作成されている。

文法規則と対になる知識として重要なもに、語彙のもつ知識がある。語彙とは単語のりのことであるが、これまで語彙の研究は、の記述量が膨大であること、地道な用例分析積み上げを長年に渡って多数行なう必要があこと等から、言語学者の多くはこれを泥沼となし敬遠して来たように思われる。

ところが最近事情が少しずつ変りつつあ言語学の新しい潮流として非変形文法の枠組が注目されている。そこでは、各語彙項目になりの情報を持たせておくことにより、これで説明が困難であった幾つかの言語現象をきいに説明し直すことが可能になることが明らになってきたからである。この新しい言語学は、語彙に書かれる知識が、言語学的な現象説明する鍵になる、という考え方に基づいてる。従来の言語学では、語彙に書かれる知識量が貧困であったために、理論的な困難さがまれて来たとする考え方である。とはいえ、語彙項目にどの様な知識をどの様な形式で記すべきかについての研究が言語学者によって分になされているとはまだいえない。

一方、自然言語処理を進める研究者は、然言語理解システムや機械翻訳システムの開を通じて、辞書にどの様な情報を盛り込むべかに依存して、システムの性能が大幅に変るとに気付いていた。このような辞書は、我々日常使う辞典とは異なり、計算機に可読な形をしていなければならない。最近我が国で電化辞書プロジェクトが始り、大規模な電子化書の構築に向けての研究開発が行なわれてい知識情報処理を背後から支えるプロジェクトして今後の成果が期待される。

［2］の常識は、高度な自然言語処理をなう場合に必要になる。このことは、例えば外の意味を理解したり、話者の意図を理解す場面を考えれば分るだろう。しかし

問題は、常識がどの様なものであるかについて、哲学者、心理学者、言語学者等によって断片的な研究はなされてはいるものの、体系的な研究がなされているとは言い難いことである。

この常識と言語に関連した問題は語用論(pragmatics)の問題であり、現在の文法理論の中心的な研究課題ではないとして、言語学者の多くはそれ以上深く追求しない。哲学者も、自然言語処理の立場から利用可能な常識という観点から常識の研究を進めて来たわけではない。最近になって、状況意味論[Barwise 83]などで、自然言語処理の研究にも有効と思われる理論が生まれつつあるが、今後のより一層の研究が必要である。

自然言語処理システムを作成する立場からは、常識の問題が困難であるからといって避けるわけにはいかない。しかし現在の研究レベルでは、この問題を一般的に解くことができる段階にはない。そこで、対象分野を極めて狭くとり、限られた範囲の自然言語処理を行なうシステムを取り上げざるをえない、というのが現状である。

3・自然言語処理と知識表現

知識表現は、人工知能の研究者によって様々な研究が行われて来た。代表的なものにセマンティック・ネットワークやフレームがある。セマンティック・ネットワークの例を図1に示す。

セマンティック・ネットワークやフレーム形式の知識表現の良さは、表現されたものの意味が直感的に理解できるという点にある。たとえば図1は次の様に読める。

(1) clyde#1の年齢は5才である、
(2) clyde#1はelephantである、
(3) elephantのcolorはgrayである、
(4) elehantはmammalである、
(5) mammalのbloodtempはwarmである、
(6) humanはmammalである。

このネットワークを使って例えば「clyde#1のbloodtempは何か」を知りたければ、isaリンクを上にたどり、bloodtempリンクを発見し、その先の値を取り出すプログラムを作成する必要がある。「何がmammalですか」という質問に答えたければ、mammalノードを発見してそこから図1のネットワークを下に辿るプログラムを作る必要がある。したがって図1のネットワークには、両方向に辿るための特別な仕掛けをあらかじめ用意しておかなければならない。

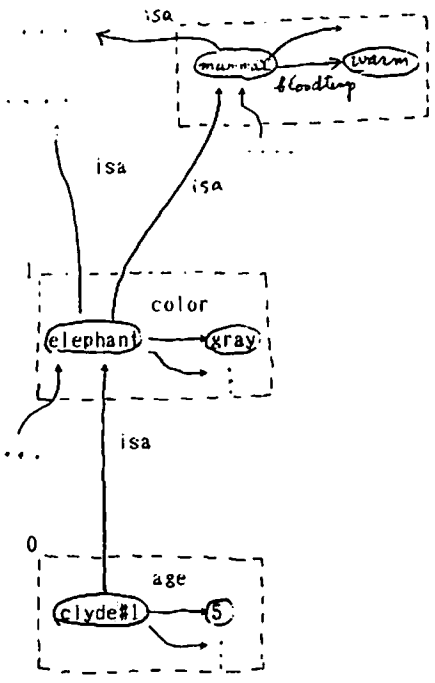


図1　セマンティック・ネットワークによる知識表現例。

このことは、推論を行なうためにセマンティック・ネットワークを自在にたどるプログラムを用意する必要があるということを意味している。こうしたプログラムを用意して推論を行なわざるをえない。

この種の知識表現形式の第2の問題点は、表現された知識の意味が使用者個々の作成したプログラムに依存することである。そのため表現されたものの形式は同じでも、その意味が使用者毎に異なることになり、外部からみて不明確になる。

第3の問題は、述語論理でいう限量記号をセマンティック・ネットワークに導入することが難しいことがあげられる。

この様な問題点があるにもかかわらず、人工知能の研究者はこれまでフレームと呼ばれる知識表現形式をよく用いてきた。そして彼らはフレーム形式とセマンティック・ネットワーク形式との間に知識表現形式としての差をしばし

ば強調する。

筆者は両者の間に大きな差があるとは考えていない。例えば図１の矩形で囲まれた部分が一つのフレームに対応していると考えられるからである。矩形は関連した知識をまとめた枠（フレーム）である。従ってフレームにも上記した意味の不明確さの問題と、フレームを解釈し推論を行なうためのプログラムを作らなければならないという問題、また限量子記号の問題がある。

図１の一部をフレーム形式（ただしPrologのリスト形式）で表現すると以下の様になる。

```
[clyde,frame,
        [isa:elephant],
        [age:5],
        ..................]
[elephant,frame,
        [isa:mammal],
        [color:gray],
        ..................]
```

フレーム形式での辞書項目の記述例を図２に示す。これは意味処理を行なう時に使う辞書項目記述例になっている。図２は意味解析の章で再び取り上げて説明する。

```
[sell,frame,
    [subj $ isa:human,agent],
    [iobj $ isa:human,goal],
    [obj $ isa:thing,object],
    [isa:action]]
```

図２ フレーム形式での辞書項目記述例

形式的な推論機構を背景にした述語論理による知識表現形式は、フレームやセマンティック・ネットワークによる知識表現に対してこれまで述べてきた問題点の全てを解消することができる。特に最近のロジック・プログラミングの枠内で知識の問題を考えることは意味がある。なぜなら、一定の形式の述語論理で表現された知識に関する推論は、全てロジック・プログラミング・システムに組み込みの機能を利用して行うことが可能になるからである。

例えばロジック・プログラミング言語の典型であるProlog上のDCKRと呼ばれる知識表現形式を用いて図２を図３の様に記述することができる。それにより意味解析のほとんど全てをPrologの組み込み機構に任せることができる。

```
sem(sell,subj:F^I^0) :-
            sem(F,isa:human),
            extractsem((agent:F)^I^0).
sem(sell,iobj:F^I^0) :-
            sem(F,isa:human),
            extractsem((goal:F)^I^0).
sem(sell,obj:F^I^0) :-
            sem(F,isa:thing),
            extractsem((object:F)^I^0).
sem(sell,P) :- isa(action,P).
```

図３ Prologによる図２の知識表現

言語学的な知識である文法規則についても、これをロジック・プログラムの形式に変換し、このプログラムに文を与えて構文解析する事も可能なことが知られている。

パターン照合が推論の基本にあることは多くの人の認めるところであろう。ところでユニフィケーションは、パターン照合の最も純粋で基本的な形態である。ロジック・プログラミングがユニフィケーションを基本計算機構としていることを考えると、述語論理に基づく知識表現は心理学的にも検討に価すると思われる。

４． 構文解析

構文解析は最も形式的で基本的な推論と関係がある。構文解析は文法を使用して、与えられた文が文法的に妥当であるかどうかを調べて、構文構造（文の主述関係、修飾／被修飾関係）を抽出するものである。
構文解析の方法には大別して３つある。

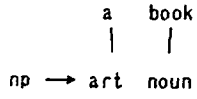
（１）前向き推論法(Forward Reasoning)
　　――これはボトムアップ法とも呼ばれる。
（２）後ろ向き推論法(Backward Reasoning)
　　――これはトップダウン法とも呼ばれる。
（３）（１）＆（２）を併用した双方向推論法
　　(Bi-directional Reasoning)。

ここで最も良く使われる文脈自由文法規則について説明する。一般に文脈自由文法規則は次の形をしている。

[1] $x \longrightarrow y \quad z$

文脈自由文法規則の特徴は、矢印の左にただ一つの記号があることである。例を挙げる；

例１）



例１）を図４に示す木構造で表すことがある。

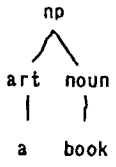


図４　構文解析木

図４は構文解析木と呼ぶことがある。この木の形から次のことが分る：

A) aが記号artに、bookが記号nounにまとめられて　いる。

B) 記号artとnounはこの順序で記号npにまとめら　れている。

したがって

C) a book は、記号artとnounとを介して、npにま　とめられている。

a book に対して文脈自由文法規則を適用することは、図４に示す構文解析木を作り出すことであると見做される。文脈自由文法規則を何度か適用して、文全体に対する構文解析木を作り上げることが、構文解析の目的であると考えることができる。作り上げた構文解析木の構造から文の主述関係、修飾／被修飾関係を知ることができる。

4・1　前向き推論法と後ろ向き推論法

一般に[1]の文脈自由文法規則には次の二通りの読みがある。矢印（$\longrightarrow$）の向きに沿った読み[1.1]とその逆向きの読み[1.2]とである。

[1.1]　解析すべき文の一部が記号xにまとめられるためには、文の一部が、記号yにまとめられる前半部分と、記号zにまとめられる後半部分とに（過不足なく）分割されるはずである。この時記号xは予測として働く（図５(a)）。

[1.2]　解析すべき文の一部の前半部分が記号yに、後半部分が記号zにまとめられることが分かると、前半yと後半zを合せて、記号xにまとめることができる（図６(b)）。

[1.1]は後向き推論法で、[1.2]は前向き推論法で構文解析する場合に使われる文法規則の読みである。これを図２と図３に示す。

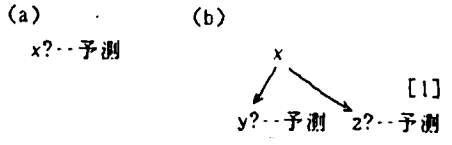


図５　後ろ向き推論法による構文解析[1.1]

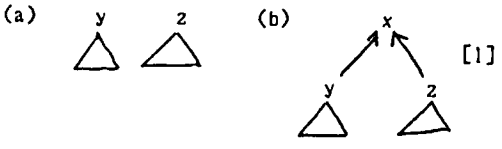


図６　前向き推論法による構文解析[1.2]

図５と図６の木に示した矢印は、構文解析木が作り上げられる方向を示している。その方向は図４では上から下に、また図５では下から上に向かっている。上をトップ、下をボトムと呼ぶことにすれば、図４がトップダウンに、図５がボトムアップに木が成長していることが分るだろう。読者は[1.1]、[1.2]の説明と、図４，５とがそれぞれどの様に対応しているかをよく見比べてみてほしい。

さてここでトップダウン法とボトムアップ法が、それぞれ後ろ向き推論法と前向き推論法に対応している理由を説明しておく。

後ろ向き推論法の特徴は、証明すべき目標が与えられると、その目標を、規則を用いてより小さな（幾つかの）副目標に分解し、次に得られた副目標に対して同様なことを何度も繰り返し、最終的に具体的事実にまで副目標が全て分解されれば、後ろ向き推論法により最初に立てた目標が達成されたことになる。通常推論の向きは、手もとにある具体的な事実から目標（結論）に向かって進むが、今の場合、目標（結論）が最初に与えられ、そこから逆向きに

具体的な事実に向けて推論が進む。そこでこれは推論の向きが普通と逆向き（後ろ向き）であると言う意味で、後ろ向き推論法と呼ばれる。

後ろ向き推論法をトップダウン法と比べてみると、図５に示す通り、まず与えられた文の部分がｘにまとめられるという目標（予測）を立て、それに[1]の文脈自由文法規則を適用して、ｙとｚという副目標を立てていることが分る。そして次にこのｙとｚとを新しい目標にして同じことを繰り返す。これは後ろ向き推論と同じ仕組の推論法であることが理解できよう。

一方ボトムアップ法では、与えられた文の単語の品詞を先ず決め、品詞の並びに対して適合する文脈文法規則を取り出し、[1]の矢印の左側の記号にまとめあげる。このような操作を何度か繰り返し、最終的な目標に文全体がまとめられれば、構文解析は成功したことになる。これは具体的な事実（文）から最終的な目標に向けて推論を進める方法で、前向き推論法とよばれている。

前向き推論法や後ろ向き推論法を用いて構文解析を行うプログラムのことをパーザ（parser)といい、構文解析することをパーズすると言うことがある。

ここで前向き推論法も後ろ向き推論法も共に必要な時点で適切な文脈自由文法規則を選択しなければならない。この時何をキーにして[1]の形の文法規則を取り出したら良いのだろうか。答えは、後ろ向き推論法の場合には、[1]の文法規則の矢印の左側にある記号ｘをキーにして取り出す。一方、前向き推論法の場合には矢印の直後の記号ｙをキーにして取り出す。

```
x ⟶ y  z
↑      ↑_前向き推論法のキー
↑_____後ろ向き推論法のキー
```

たとえば後ろ向き推論法の場合、図２を参照すると、目標（予測）ｘを立てると、矢印の左に記号ｘを持つ規則を取り出して副目標ｙとｚとに展開していることが分る。したがって後ろ向き推論法の場合、ｘをキーにして文法規則を取り出すことができるように、文法規則を配列しておくと都合が良い。

読者は図６を眺めて、前向き推論法の場合に、[1]の矢印の直後の記号ｙをキーにして文法規則を取り出すと都合が良い理由を考えてみてほしい。前向き推論法による構文解析を行うためには、[1]の（矢印の直後の）記号ｙをキーにして文法規則が取り出せるよう文法規則を配列しておく。

最後にそれぞれの方法の特徴を説明しておきたい。後ろ向き推論法による構文解析の最大の欠点は、左から右に向けて文の単語を取り出して構文解析を進める場合に、[1]の矢印の左の記号ｘが矢印の直後の記号ｙと等しい文法規則があると、構文解析が止らなくなることである。これを左再帰型文脈自由文法規則とよぶ。左再帰型文脈自由規則を適用すると、後ろ向き推論（トップダウン）法による構文解析が止らなくなる様子を図７に示す。

左再帰型文脈自由文法規則：

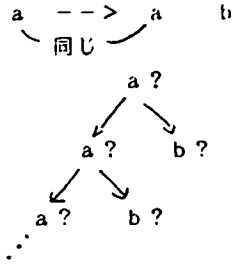
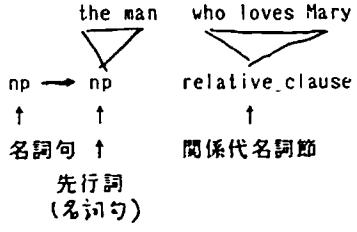


図７　左再帰型文脈自由文法規則と無限ループ

事実、日本語や英語の文法を文脈自由文法規則で書いてみると、左再帰型の規則が必要になることが多い。たとえば英語の関係代名詞文に対する文法規則を書こうとすると、以下のような左再帰型の文法規則になる。



一方、前向き推論法による構文解析では左再帰型の文法規則があっても無限ループに入り止らなくなることはない。そのため自然言語処理の分野では、前向き推論法による構文解析をおこなうことが多い。

後ろ向き推論法による構文解析の利点は、

予測に導かれた解析を行うことができることである（図５、図６参照）。予測が精密になればなるほど、予測に合わない解析結果を早期に排除することができるので無駄な解析をしなくてすむ。そのため後ろ向き推論法のことを、予測（仮説）に導かれた推論法という意味で、仮説駆動型推論法と呼ぶことがある。

前向き推論法による構文解析では、構文解析木が下から上に向けて成長する時、一体この木が所望の目標に向かって成長しているのかどうかが解析中に分からないのが問題である。

４・２　双方向推論法

双方向推論法による構文解析について説明する。前節では構文解析には、エキスパート・システムで用いられる前向き推論法と後ろ向き推論法にそのまま対応する方法があることを説明した。そしてそれぞれに長所短所が有ることを説明した。それでは、両者の長所を生かした効率的な構文解析法はあるのだろうか。幸いにしてそれは存在する。それが双方向推論法である。

前節の最後に述べたように、後ろ向き推論法による構文解析では、左再帰型の文法規則を適用すると無限ループに入り止らなくなることがある。一方前向き推論法による構文解析では、予測に導かれた解析を行うことができない。

そこで双方向推論法による構文解析では、前向き推論法を基本にし、しかも後ろ向き推論法をも用いて、下から上に向けて成長する木の内、予測に合わないものを早期に排除する。以下ではそれがどの様にして可能になるかを説明する。解析は文の左から右に向けて進むものとする。

まず双方向推論法による構文解析では、予測としてｓ（sentence;文）を立てる。これは文の解析が全て終った段階で、文がｓを根とする構文解析木にまとめられなければならない、という予測になっている。さてこの時、文の一部が記号ｙにまとめられ、ｙから左に向けて解析が進むとしよう。木は下から上に向けて成長するはずであるが、その木が予測ｓに至る可能性があるということがあらかじめ分っているものとしよう（図８（ａ））。実はｙからｓに至るかどうかをどの様にして予め知るかが問題であるか、これについては後で詳しく説明する。

そうすると、前向き推論法により文脈自由文法規則の矢印の直後に記号ｙがある規則を取り出す。たとえばそれが２・１の[１]であったとすると、それを適用して図５（ｂ）を作る前に、記号ｘから上に向けて木が成長した時予測ｓに到達する可能性が有るかどうかを調べる。もしその検査に合格すると図５（ｂ）を作り出す。

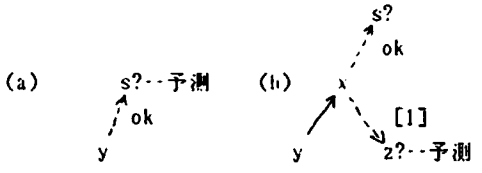


図８　双方向推論法による構文解析

図８（ｂ）は、記号ｙにまとめられた部分に直接後続している残りの文の部分が、記号ｚにまとめられる筈だと言う（新しい）予測ｚを立てたことになる。以下同様にして、この新しい予測ｚに対して双方向推論法による構文解析が繰り返される。以上が双方向推論法による構文解析の基本的な考え方である。

ここで到達可能性をどの様にして知るか、その方法を説明しよう。問題は「予測Ａがあり、Ｂを根とする木があるものとしよう。解析が更に右に進み、Ｂから上に木が成長した時、それが予測Ａに到達可能かどうかをどの様にして知るか」というものである（図９参照）。

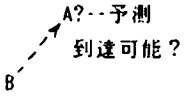


図９　到達可能性の検査

幸いにして到達可能性に関する情報は、文脈自由文法規則が与えられた段階で知ることができる。これは次の様に述べることができる。

文法規則の系列：

```
A1 --> B     C1
A2 --> A1    C2
    .......
An --> An-1  Cn-1
```

A ⟶ An　Cn

があれば、構文解析木上で、BからAに到達可能な経路がある。但し、Ci(1≦i≦n)は省略されていても良い（図１０）。

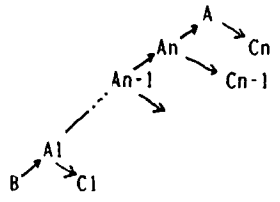


図１０　構文解析木上で到達可能な経路

以上のようにして、文脈自由文法規則の集合が与えられた段階で、文法規則に現われる全ての記号対に対して、それが到達可能な記号対であるかどうかを予め計算しておくことができる。この記号対を用いて、下から上に向かって成長する木が将来予測を満たすかどうかを決めて、無駄な木の成長を早期に止めることができる。この様にして双方向推論法による効率の良い構文解析が可能となる。

５．　意味解析

５・１　意味処理の基本的な考え方

意味解析の基本となるのは、辞書項目の記述である。辞書項目の記述として良く用いられるものとしてフレーム表現がある。図２にその一例を示したが、それをここで繰り返して示す。

```
[sell,frame,
    [subj $ isa:human,agent],
    [iobj $ isa:human,goal],
    [obj $ isa:thing,object],
    [isa:action]]
```

図２のフレームを用いた意味解析を説明する。この方法は、動詞を中心にした意味解析であるが、動詞が与えられると、その動詞がどの様な名詞句と共起するかを容易に予測することができる。この予測を用いて意味解析を効率良く進めることができる。

フレームはスロットの集合からできている。図２のフレームは４つのスロットから構成されている。最初のスロットは、スロット名(subj)、制約条件(human)、アクション(=agent)の３組みからなる。ここで制約条件は、このスロットを満たすことのできるフレーム（これをフィラーという）の意味的性質を記述する。
ここで動詞のフレームのスロットのフィラーとなるものは動詞と共起する名詞であることに注意。

例えばフィラーがこのスロットを満たす（埋める）ことができるためには、フィラーが文中で、主格(subj)の位置を占めhumanでなければならない、ということが記述されている。フィラーがこのスロットを満たすとアクション(=agent)を起動し、フィラーの深層格をagentであるとする。フィラーがこのスロットを満たすことができなければ、つぎのスロット(iobj)を選択し同様なことを行う。フィラーの満たしうるスロットが発見できなければ、selfスロットを介して上位のフレームactionの持つスロットにアクセスする。

[田中 86a]によると、DCKR[小山 85]とよばれる形式で辞書項目を記述しておけば、以上に述べた意味解析の手順のほとんど全てをPrologの基本計算機構に任せることができる。これについて簡単に説明する。

DCKRでは、フレームを構成する各スロットを、
sem 述語をヘッドとする一つのホーン節で表現する。ただし、このsem 述語の第一引数にはフレーム名がくるものとする。図２の記述をDCKRで表現したものを図３に示した。ただし図３の中で上位概念をたどるための述語isaの定義を以下に示す。

```
isa(Upper,P) :-
        P=isa:Upper;
        sem(Upper,P).
```

さて図３の各ホーン節が図２の各スロットに対応している。スロット名は、ホーン節のヘッドの第２引数の先頭に書く。変数Fがフィラーである。スロットに対応するホーン節のボディーには、意味的制約条件とアクションの対（以後これをＣＡ対とよぶ）が記述されている。たとえば図３の最初のホーン節のボディーの

```
sem(F,isa:human),
extractsem((agent:F),In,Out)
```

という記述は、フィラーFが人間（human）なら、フィラーFの深層格を動作主(agent)とするアクションextractsem((agent:F),In,Out)を起動し、深層格構造を抽出するものである。ここで、フィラーFが人間であるかどうかを調べるsem(F,isa:human)はフィラーFに対する意味的制約条件を表わしている。extractsemは、抽出した深層格構造をInに付加した結果をOutに返す。

仮にフィラーがmaryであるとして、

```
sem(sell,iobj:mary^|^0)
```

を実行して図３の記述を呼出せば、Prologの基本計算機構によりスロットの選択が自動的になされる。今の場合Prologは図３のヘッドのうちスロット名がiobjであるものを選択する。次にそのボディの実行に移る。

ボディではまず

```
sem(mary,isa:human)
```

を実行するが、これは成功する。そこで次にextractsem((goal:mary)^|^0)を実行して、maryがsellの深層格goalであるとする意味を抽出することができる。

もしフィラーをmonday、スロット名をtimeとして、

```
sem(sell,time:monday)
```

を実行すると、図３の初めの３つのスロット（ホーン節）をPrologは調べるが、スロット名が異なるので最後のホーン節を選択する。そしてsellの上位のactionにあるスロットを自動的にアクセスすることになる。
これは知識継承がPrologのユニフィケーション機構により自動的になされる例になっている。

このように、辞書項目記述形式をDCKRにすれば、意味解析の中核をなすプログラムをPrologに組み込みの基本計算機構で全て代用可能なことが分かる。

さらに図３の記述はコンパイルすることができるので、高速化を図ることができる。これまでの辞書項目の記述形式では、辞書項目を一つの大きなデータ構造として表現していたため、それをコンパイルできないのと良い対照をなす。

意味解析結果をもDCKR形式にしておけば、質問文から得られたDCKR形式の結果をそのまま実行して答えを得ることができる。例えば

```
John sold Mary a book.
```

という文から

```
sem(sell,agent:john),
sem(sell,object:book),
sem(sell,goal:mary),
sem(sell,tense:past),
  ..............
```

を抽出しておけば

```
To whom John sold the book?
```

という文から、

```
sem(sell,agent:john),
sem(sell,object:book),
sem(sell,goal:ToWhom),
sem(sell,tense:past),
  ..............
```

が抽出して、単にこれを実行すると

```
ToWhom = mary
```

という答えをPrologは返すことができる。この場合Prologインタープリタがそのまま推論エンジンとして働く。

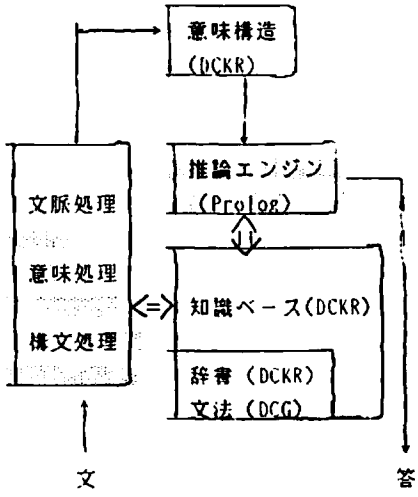


図１１ DCKRと自然言語理解システム

以上を考慮すると、Prologをベースにした自然言語理解システムの構成として図１１が得られる。図11の左側には言語処理を行なう部分があり、図１１の中心に知識ベースがある。文が入力されると、知識ベースを用いて構文処理、意味処理、文脈処理が行なわれる。網目をかけた部分はPrologの基本計算機構で置き換え可能なことを意味している。

論理型プログラミング言語に関連した自然

言語処理関連の文献を中心に以下に示す。筆者は論理型プログラミング言語による自然言語処理の新しいパラダイムが始まっていると考えている。この方向の研究としてコンピュータソフトウエアの１０月号の「論理と自然言語」特集号を参考にされたい。また情報処理の８月号の「計算言語学」特集号も参考になる。

参考文献

[Barwise 83] Barwise,J. and Perry,J.: Situations and Attitudes,The MIT Press (1985).

[Bobrow 77] Bobrow,D.G.et.al.: An Overview of KRL-0, Cognitive Science,1,1, 3-46 (1977).

[Clocksin 81] Clocksin,W.F.et.al.: Programming in Prolog, Springer-Verlag, (1981).

[Colmeraure 78] Colmeraure,A.: Metamorphosis Grammer,in Bolc (ed): Natural Language Communication with Computers, Springer-Verlag,133-190(1978).

[淵 77]：述語論理的プログラミング－－EPILOGの提案、情報処理学会第１回記号処理研究会資料(1977.7).--情報処理,26,11,1298-1306(1985)に再録.

[小山 85] 小山晴生（他）：Definite Clause Knowledge Representation,Proc.of LPC'85, ICOT,95-106(1985).

[Matsumoto 83] Matsumoto,Y. et.al.: BUP--A Bottom-up Parser Embedded in Prolog, New Generation Computing, 1, 2,145-158 (1983).

[松本 86] 松本裕治：並列構文解析、情報処理学会自然言語研究会資料、(1986).

[Mellish 85] Mellish,C.S.: Computer Interpretation of Natural Lnguage Descriptions, Ellis Horwood Limited(1985).

[Mukai 85] Mukai,K.: Unification over Complex Indeterminates in Prolog, Proc. of LPC'85,ICOT,271-278(1985).

[Pereira 80] Pereira,F. et.al.: Definite Clause Grammar for Language Analysis --A Survey of the Formalism and a Comparison with Augmented Transition Networks, Artificial Intelligence,13, 231-278(1980).

[Pereira 81] Pereira,F.: Extraposition Grammar, American Journal of Computational Linguistics,7,4,243-256 (1981).

[Shieber 86] Shieber,S.M.:An Introduction to Unification-Based Approaches to Grammar,The Center for the Study of Language and Information,Stanford University, 1986.

[田中 86a] 田中穂積（他）：知識表現形式DCKRとその応用、コンピュータソフトウエア、日本ソフトウエア学会編、岩波書店（掲載予定）.

[田中 86b] 田中穂積（他）：自然言語処理のためのソフトウエアシステムLangLAB、Proc. of LPC'86,ICOT, 5-12(1986).

[Winograd 83] Winograd,T.:Language as a Cognitive Process,vol.1:Syntax,Addison-Wesely(1983).