2A-2

オブジェクトの同一性判定及び同一化アルゴリズムとその応用

A Method of Semantic Matching

田中穂積、奥村学、小山晴生（東京工業大学　工学部）

シンタクス情報だけでなく、意味を考慮した２つのオブジェクト同士の同一性判定及び同一化アルゴリズムについて述べ、その自然言語処理への応用についてもふれる。

1．はじめに

述語論理式で表現した知識と、フレームもしくはStructured Object で表現した知識との間の関係は［Hayes 80］,［Nilsson 80］,［Goebel 85］,［Bowen 85］等により明らかにされている。そのなかで［Nilsson 80］は、述語論理式で表現した知識を一度Structured Object 形式に変換し、それを意味ネットワーク形式で表現する方法をのべている。それによれば、Structured Object に関する推論は、それと等価な意味ネットワークをたどる操作に還元される。したがって、推論／問題解決をおこなうためには、別途この意味ネットワークに対するインタープリータを作る必要がある。

それにたいして筆者らは、DCKR(Definite ClauseKnowledge Representation) とよぶ知識表現形式を開発している［小山　85］。DCKRはホーン節形式をベースにした知識表現形式であり、意味ネットワーク形式の知識をDCKRで容易に記述することができる。DCKRでは、Stuructured Objectを構成する各スロットを、sem 述語（後述）をヘッドとする一つのホーン節（一つのProlog文）で表現する。したがって一つのStructured Object は、第一引数が等しいsem 述語をヘッドとするホーン節（スロット）の集合としてみなすことができる。以上のことからDCKRで記述した知識に関する推論を行なうプログラムは、Prologに組み込みの機能で代用することができることが分かる。推論を行なうために特別なプログラムを作る必要はないのである。

人工知能の分野では様々なStructured Object （以後これを単にObjectとよぶ）を扱わねばならない。この時 Object 相互間のパターン．マッチを行なう必要がしばしば生じる。この方向を更におしすすめて、Object相互間のユニフィケーション（同一化）を基本計算機構に取り入れた言語 CILが開発されている［Mukai 85］。

Object 相互間の同一化を行なう場合の一つの問題は、Objectを構成するスロットの並びに関する順序的な制約がないため、それらを考慮した同一化が必要になる事である。これは、Objectを一つの大きなデータ構造（例えばリスト構造）として表現した時には計算コストがかかるので問題となる。DCKRを用いればスロットは一つのホーン節として表現されるため、同一化時に必要になるスロットの選択はPrologに組み込みのバックトラックの機能に任せることができ問題は若干軽減する。それでも、Object相互間の同一化を基本計算機構に組み入れた言語を設計する場合には、CIL のように同一化は、シンタクスのレベルに留めざるをえないだろう。

一方、意味を考慮したObject間の同一化の重要性は、［Bobrow 77］ではForced Matching として、［Nilsson 80］ではSemantic Matching として、またPrologのユニフィケーションを拡張する試みとして論じられている。以下では２章で述べる比較的単純な言語的な知識を背景にして、Object相互間の同一化を、DCKRをベースにして行なう方法を述べる。これは長期を要する研究課題であることはよく知られている。したがって以下で述べる方法は、この問題の解決に向けてのささやかな一歩であるに過ぎない。

2．DCKRによる知識表現

第３章で使うDCKRによる知識表現例を以下に示す。

前節で述べたようにsem 述語の第一引数は、Objectの名前になっている。Objectの名前の中で＃の付いたものは「個体」を表わす。＃の付かないものは「プロトタイプ」を表わす。

```
    :-op(100,yfx,~),
      op(100,yfx,:),
      op(90,xfy,#).
01 sem(moore#1,adviser:mcCarthy#1,_).
02 sem(moore#1,P,S) :-
        isa(csStudent,P,[moore#1|S]).
03 sem(csStudent,P,S) :-
        isa(human,P,[csStudent|S]).
04 sem(human,P,S) :-
        isa(mammal,P,[human|S]).
05 sem(elephant,P,S) :-
        isa(mammal,P,[elephant|S]).
06 sem(mammal,bloodTemp:warm,_).
07 sem(mammal,P,S) :-
        isa(animal,P,[mammal|S]).
08 sem(animal,P,S) :-
        isa(creature,P,[animal|S]);
        hasa(body,P,[animal|S]).
09 sem(creature,age:X,S) :-
        bottomof(S,B),
        sem(B,birthYear:Y,_),
        X is 1985 - Y.
10 sem(clyde#1,age:5,_).
11 sem(clyde#1,P,S) :-
        isa(elephant,P,[clyde#1|S]).
12 sem(elephant#1,birthYear:1980,_).
13 sem(elephant#1,P,S) :-
        isa(elephant,P,[elephant#1|S]).
14 sem(elephant#2,birthYear:1982,_).
15 sem(elephant#2,P,S) :-
        isa(elephant,P,[elephant#2|S]).
16 sem(mcCarthy#1,address:stanford,_).
17 sem(mcCarthy#1,
                nationality:american,_).
18 sem(mcCarthy#1,
        affiliation:stanfordUniversity).
19 sem(mcCarthy#1,P,S) :-
        isa(human,P,[mcCarthy#1|S]).
20 sem(mister5G#1,address:japan,_).
21 sem(mister5G#1,P,S) :-
        isa(human,P,[mister5G#1|S]).
22 sem(misterAI#1,address:america,_).
23 sem(misterAI#1,P,S) :-
        isa(human,P,[misterAI#1|S]).
24 sem(america,capital:newYork,_).
25 sem(america,climate:temperate,_).
26 sem(america,P,S) :-
        T = [america|S],
        isa(coutry,P,T);
        hasa(california,P,T);
        ..............
27 sem(california,P,_) :-
        T = [california|S],
        isa(state,P,T);
        hasa(stanford,P,T);
        ..............
```

01の記述は、moore#1という（ユニークな名前を持つ）個体のadviserがmcCarthy#1であるという事実を示している。ここでadviserは「スロット名」、mcCarthy#1はadvidserというスロットの「値」であるという。02の記述はmoore#1はcsStudent（計算機科学科の学生）であるという事実を示している。ここで02は上位からの知識の継承をそのまま記述したものである事に注意したい。02は、csStudentが性質Pを持てばmoore#1もその性質Pを持つと読むのである。isaは、Objectの階層をたどるための述語であり、その定義をhasaとともに以下に与える。

```
28 isa(Upper,P,S) :-
        not(member(Upper,S)),!,
        Prop = isa:Upper;
        sem(Upper,P,S).
29 hasa(Part,X:Y,S) :-
        X == hasa,
        not(member(Part,S)),!,
        Y = Part;
        sem(Part,hasa:X,S).
```

sem述語の第3引数は、階層をたどった道筋をスタックし、isa述語のボディーの初めで、たどった道筋がループしているかどうかをチェックするために使われる。

03-04はcsStudentがhumanであり、humanがmammalであるという事実を記述している。08はanimalはcreatureでありbodyを持っているという事実を記述している。09は呼び出し元（B）のbirthYearが知れている時、それを用いてageを計算する知識である。たとえば

```
?-sem(elephant#1,age:X,_).
```

を実行すると、isa述語の働きによって09に至り

```
X = 5
```

を得ることができる。また

```
?-sem(elephant#1,P,_).
```
を実行すると
```
P = birthYear:1980;
P = isa:elephant;
P = isa:mammal;
P = bloodTemp:warm;
P = isa:animal;
P = isa:creature;
P = age:5;
```
のようにelephant#1に関する知識を次々に得ることができる。

上位から下位の概念をアクセスしたければ、単にsem(X,isa:mammal,_) を実行すればよい。述語hasaの機能を調べるために、読者は
```
?- sem(america,hasa:X,_).
```
を実行して見るとよい。

これまでの説明から10以降の記述がどのようなものであるか理解できるであろう。また推論を行なうプログラムがほとんど不要であることが分かるだろう。知識をDCKRで記述しておきさえすれば、推論はPrologに組み込まれたインタープリータによって自動的になさるからである。DCKRにより記述された知識は読みやすいと思われるがどうであろうか。このことは、知識の記述のしやすさにも通じる。

以上の他に、知識の体系に関する次のメタ知識がある。それらの使い方は３章で説明される。
```
30. metakb(mammal,ntype:exclusive).
31. metakb(age,ltype:exclusive).
32. metakb(address,ltype:exclusive).
```

３．Object相互の同一性を判定するアルゴリズム　－－Semantic Matcher

２章のDCKRの記述からmcCarthy#1のaddress はstanfordで、misterAI#1のaddress はamerica であることがわかる。我々はstanfordがamerica のなかにあるという事実を知っているから、misterAI#1とmcCarthy#1とを同一視してもなんら矛盾はないと推論することができる。一方同様にして、america とjapan は異なる国であるから、mister5G#1とmcCarthy#1とを同一視する事はできないと推論することができる。このように意味を知り２つのObjectの同一化を行なう操作をSemantic Matching(Forced Matching)とよび、Semantic Matchingを行なうプログラムをSemantic Matcherとよぶ。

これまで、Semantic Matchingの必要性はしばしば論じられてきたが、Semantic Matcherを作成したという試みは皆無と言ってよい。それは、上記した程度のレベルのSemantic Matcherの作成でさえ、相当複雑なものになると考えられたからであろう。しかし、DCKRによる知識表現によれば、先に述べた程度のSemantic Matcherは容易に作成することができる。そこで用いられるアルゴリズムの概要を以下に示す。

[A]　同一化の対象になる２つのObject (o#1とo#2) に共通の上位Object (Oi) があればそれを取り出し [B] へ、さもなければ [D] へ。

[B]　Oiの直接１レベル下位のObjectが互いにexclusive な関係にあるという事がわかれば（これはmetakb(Oi,ntype:exclusive) が成り立つかどうかを調べることによって、容易に知ることができる（２章））[C] へ、さもなければ [A] へ。

[C]　Oiの直接１レベル下位に相異なる２つのOj,Okがあり、それぞれ o#1とo#2の上位に位置しているなら、同一化は失敗であるとしてリターン。さもなければ [A] へ。

[D]　o#1とo#2のおののに共通のスロット名をもつものがあれば、それらを全部対にして取り出し [E] へ。一つもなければ、o#1とo#2とを同一化してはならないという積極的な理由が発見できないとみなし、同一化が可能であるとしてリターン。

[E]　取り出されたスロットの対(Ax:Bx,AX:By) の集合から、次のような新しい集合Ｓをつくる。
```
S = {(Ax:Bx,Ax:By)
        | metakb(Ax,ltype:exclusive)}
```
この時集合Ｓの全ての要素に対して
```
Bx == By;
sem(Bx,isa:By,_);
sem(By,isa:Bx,_));
sem(Bx,hasa:By,_);
sem(By,hasa:Bx,_)
```
が成り立てば同一化が可能であるとしてリターン、さもなければ失敗であるとしてリターン。

[F]　[A] から [E] のアルゴリズムによ

り、o#1とo#2の同一化が可能な場合には、o#1とo#2とを等しいとおくために、次の事実をassertする。

```
sem(o#1,P,S) :-
    isa(o#2,P, [o#1|S] ).
sem(o#2,O,S) :-
    isa(o#1,O, [o#2|S] ).
```

以上によりo#1 にはo#2の、またo#2にはo#1の性質が自動的にうけつがれる。

4．実行例

[A] から [E] のアルゴリズムはおよそ40行のmkeqと呼ばれる述語により実現されている。以下に実行例を示す。

```
(a) ?- mkeq(x#1,y#1).
(b) yes
(c) ?- mkeq(x#1,misterAI#1).
(d) yes
(e) ?- sem(y#1,P,_).
(f) P = isa:x#1;
(g) P = isa:misterAI#1;
(h) P = address:america;
(i) P = isa:mammal;
(j) P = bloodTemp:warm;
(k) P = isa:animal;
(l) P = isa:creature;
(m) no
(n) ?- mkeq(mcCarthy#1,mister5G#1).
(o) no
(p) ?- mkeq(mcCarthy#1,misterAI#1).
(q) yes
(r) ?- mkeq(mcCarthy#1,clyde#1).
(s) no
(t) ?- mkeq(elephant#1,clyde#1).
(u) yes
(v) ?- mkeq(elephant#2,clyde#1).
(w) no
```

(a)ではx#1とy#1という２つのObjectを作り出し、それらを等しいと置いている [D]。(c) ではx#1とmisterAI#1とを等しいと置いている [D]。したがって(e)でy#1のもつ性質を聞くと(f)-(l)の応答に見るように、y#1はmisterAI#1の性質を受け継いでいることが分かる。(n),(p),(t),(r)の応答は [E] によっている。ここで(t) と(v)では、elephant#1とelephant#2のいずれも陽にage についての記述がないにもかかわらず、正しい応答を示している事に注意したい。その理由は２章で述べた。(r)にたいする応答は [C] による。mcCarthy#1もclyde#1も、共にmammalを共有するが、その直下にあるhumanとelephantはexclusiveな関係にあるからである。

5．応用

複数の文の連鎖からなる談話を理解するためには、前文を受けながら談話が進行する過程で様々なObjectが発生し、それらうちどれとどれが同一であるかを推論することが必要になる。例えば、言語学でいう前方照応がその典型である。本稿で述べたSemantic Matcherは、この問題に対して部分的に応用可能だろう。また、人工知能の分野でこんご重要になると思われるAnalogical Reasoningやそれをさらに深めた学習の問題などにも応用可能だと思われる。

6．おわりに

DCKRによる知識表現形式を用いることにより本稿で説明したレベルのSemantic Matcherを容易に実現することができた。これを自然言語理解システムへ応用することを考えていきたい。

7．参考文献

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

[Bowen 85] :Meta-Level Programming and Knowledge Representation,Syracuse Univ.,(1985).

[Goebel 85] :Interpreting Descriptions in a Prolog-Based Knowledge Representation SystemProc.of IJCAI'85,711-716(1985).

[Hayes 80] :The Logic of Frame,in Frame Conceptions and Text Understanding, Walter de Gruyer, Berlin,46-61(1980).

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

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

[Nilsson 80] :Principles of Artificial Intelligence,Tioga, (1980).

2A-3　トップダウン法とボトムアップ法を融合した新しいパージング方式

A New Parsing Method Mixed by Top-Down and Bottom-Up Strategies

上脇　正、田中穂積（東京工業大学・工学部）

論理プログラミングの枠組みで、トップダウン法とボトムアップ法とを融合させた新しいパージング方式を考案したので、そのアルゴリズムと、文法規則をトップダウン規則とボトムアップ規則に分類する方法を述べる。

１．はじめに

トップダウンによる解析は、解析に必要になる予測情報ができるだけでなく、Prologの問題解決の戦略とも一致するため［Pereira 80］、Prolog上に簡単に高速なパーザを実現できる。しかし、トップダウン縦型探索による解析には、左再帰的規則があるとそこで無限ループに入るという欠点がある。一方、ボトムアップによる解析は左再帰の規則を扱えるが、予測を用いた解析が行えない。

従来の方式では、パーザが一つの書き換え規則をトップダウンの向きに解釈したり、ボトムアップの向きに解釈したりして解析を行うものであった。この方法によると、パーザが、どのタイミングでトップダウンかボトムアップの向きに規則を解釈すべきかが明確ではないため、必ずしも有効な方法ではなかった。

本論文では、トップダウン法とボトムアップ法の長所をできる限り取り入れるために、トップダウン法とボトムアップ法を融合した新しいパージングの方式を提案する。本方式では、文法規則をトップダウンに解析を行うために用いる規則とボトムアップに解析を行うために用いる規則とを明確に分類する。トップダウン規則はPereira の方法で解析を行ない、ボトムアップ規則はＢＵＰ節［松本83］を用いる。トップダウン法とボトムアップ法を融合したパージングは、トップダウン規則とボトムアップ規則が相互に呼出し通信し合いながら進行する。したがって、トップダウン規則とボトムアップ規則とが相互に呼合う機構が必要である。これについては、第２章で述べる。本論文で提案するトップダウン法とボトムアップ法の融合パージングではどのタイミングでトップダウンからボトムアップ、ボトムアップからトップダウンの構文解析に切り替えるべきかは、各文法規則が知っていることになる。このような文法規則の自動生成法の一つの方法を第３章で述べる。

４章でパージングの高速化について述べる。

図1-1がシステムの概略である。ユーザはＤＣＧを

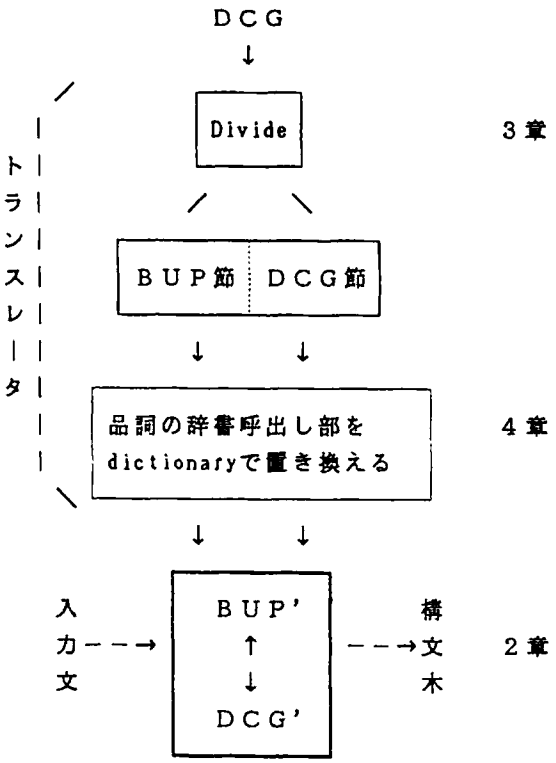

図1-1トップダウン法、ボトムアップ法融合パーザ

入力するとトランスレータによりＤＣＧはボトムアップ法で解析を行う規則とトップダウン法で解析を行う規則に分類され、それぞれＢＵＰ節とＤＣＧ節に変換される。その結果に更に高速化のための変換が行われる。解析時は両方の規則が互いに呼合いながら構文木を生成する。

２．トップダウン規則とボトムアップ規則の相互呼び出し

まず、'The boy walks' を例にトップダウン法とボトムアップ法による解析の順番を比べ、その後で呼出し機構の説明をする。

図2-1 がPereira の方法のトップダウンによる解析の順番である。縦型探索の典型的な順番の上から下、左から右の順番に解析が進む。

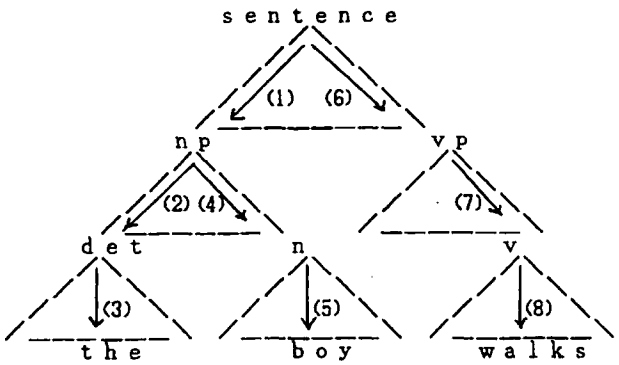

図2-1 トップダウンの解析順序

一方、図2-2 はＢＵＰのボトムアップによる解析の順番である。最初に辞書引きを行ない、それを手がかりに解析が進められる。このため左下から段々に部分木が形成され最後に全体の構文木となる。ただし、ＢＵＰでは、解析の効率を上げるためにトップダウン的予測を用いている。このため３、６では下向きに予測がなされている。つまり、６の場合、npの部分木が１－４により出来上がると、文法規則の中より右辺の先頭がnpである規則

```
sentence --> np,vp.
```

を捜し出し、その規則によりnpの右にvpが来ることを予測している。この予測の部分が６である。
そして、８により出来上がる部分木（この部分木を作るのにも６の予測が使用される。）が、６の予測と一致するとこの部分の解析が成功したことになる。

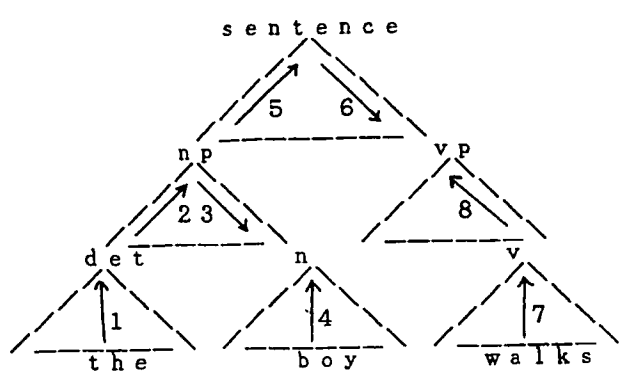

図2-2 ボトムアップの解析順序

３の部分も同様である。

このトップダウン的に予測されたカテゴリを利用するとボトムアップ規則とトップダウン規則で相互に呼び合うことが可能となる。

まず、ボトムアップ規則からのトップダウン規則の呼び出しを考えてみる。図2-2 で６の部分はトップダウン的予測を行なっている。即ち、この段階で既にその下に来るべきカテゴリ(VP)が分かっていることになる。よって、８による下からの部分木と６の予測との一致を見る代わりに、６で予測されたカテゴリを用いて、図2-1 の(7),(8) のようにトップダウンに解析を行なうことが可能である。このようにしてに、ボトムアップ法からトップダウン法への切替がなされる。

実際のシステムでは、図2-1 の(6) の部分のＤＣＧの呼び出しは、

```
vp(Arg1,Arg2)
```

のようになっている。一方、ＢＵＰに於ける図2-2 の６の部分の予測付のボトムアップ規則の呼び出しは、goalという節を用いて次の様になっている。

```
goal(vp,[Arg1,Arg2])
```

第１引数が予測されるカテゴリである。これらは解析手順は異なるが、全く同じ結果を得ることが可能であり、相互に入れ替えても支障がない。ＢＵＰ節中のgoal節をＤＣＧ節の呼び出しに置き換えることにより、ボトムアップ規則からトップダウン規則の呼び出しが可能である。

例：
```
np(G,0,Info) --> [link(np,G)],
                 goal(vp,0),
                 sentence(G,0,Info).
                   ↓
np(G,0,Info) --> [link(np,G)],
                 vp,
                 sentence(G,0,Info).
```

トップダウン規則からボトムアップ規則を呼ぶ場合は、この逆を行なえばよい。図2-1 の(6) のカテゴリ(vp)が予測されたとして、図2-2 の７、８のようなボトムアップの解析を行ない、(6) と８の解析の結果を比較する。実際のシステムでは、ＤＣＧの規則の中にgoal節を入れることにより、トップダウン規則からボトムアップ規則の呼び出しが可能になる。

例：
```
sentence --> np,vp.
               ↓
sentence --> np,goal(vp,0).
```

3．ボトムアップ規則と
トップダウン規則への分類

ボトムアップ規則とトップダウン規則の相互呼び出しを行なうためには、文法規則を予めボトムアップ規則とトップダウン規則に分けておく必要がある。人間がこの分類を行うのは大変なので、ある種の基準を設けて計算機に自動的に規則の分類をやらせることにする。この章ではその分類を自動的に行うアルゴリズムについて述べる。

まず、トップダウン規則とボトムアップ規則に分けるための基準を述べる。我々は以下に示す３つの基準を考えている。

基準１：スタート・シンボルはトップダウンに呼び出す。スタート・シンボルはトップダウン規則かボトムアップ規則のどちらかに確定しておかなければ、解析を開始することが出来ない。ここではスタート・シンボルはトップダウン規則で書くと約束する。

基準２：トップダウン規則は、左再帰の規則を含んではいけない。（ここの左再帰の規則とはa --> a,b,c. のような直接的な再帰だけでなく、a --> b.　b --> c.　c --> a.のような再帰も含む。）１章で述べたように、トップダウン＆縦型探索で解析を行なう場合、左再帰の規則があると無限ループに入る。

基準３：ボトムアップ規則からトップダウン規則を呼ぶ場合、規則の右辺の最初の文法カテゴリを呼び出してはならない。図2-2 を見ると２を行なうためには、その下の部分木が完成していることが必要である。なぜなら、１の部分木が完成してなく、この部分のカテゴリが分からないと２、３でどの文法規則を使用するかが決定できないからである。下の部分の完成により上の部分の解析が始まるということはトップダウンの規則が使えないことを意味する。

以上の基準により、次のような分類アルゴリズムが考えられる。

1. 文法規則の左辺のカテゴリclと右辺の最初の要素crの関係をcl->crと表わす。

例：　sentence --> sdec,conj,sdec.
ならsentence -> sdec

2. sentence -> sdec -> subj　のような関係がある時、１回以上の'->'の関係により到達可能な関係を

sentence => sdec
sentence => subj
sdec　　=> subj

のように表す。

3. カテゴリの集合C_loopを次のように定義する。

C_loop = {c | c => c}

この集合の要素は全てトップダウン法により解析を行おうとすると無限ループに陥るカテゴリである。

4. カテゴリの集合C_bottomを次のように定義する。この集合の要素についての規則はボトムアップ規則にする。

C_bottom = { c | cl => c, cl∈C_loop}

5. C_bottomの要素以外のカテゴリはトップダウン規則であるとする。

ただし、スタート・シンボルは予め無限ループにならないように文法規則を作成しておく必要がある。

以上に述べた文法規則の分類アルゴリズムを用いてトランスレータは、与えられたＤＣＧの文法規則を自動的にトップダウンとボトムアップに分類し変換する。（図1-1参照）

4．パージングの高速化手法

4.1. 品詞の直接辞書引き

２章で述べたようにＢＵＰではgoalという節を解析に用いている。この節は、

```
goal(G,A,X,Z) :- dictionary(C,A1,X,Y),link(C,G),
                 P=..(C,G,A1,A,Y,Z),call(P).
```

のようなものであり、辞書引きを行い、それを使って上へ木を伸ばしている。ＢＵＰ節に変換された規則の高速化をはかるために次のような性質を満たすカテゴリ(Hinsi)を予測に持つgoal節の呼出しを直接dictionary節の呼出しで置き換えることができる。

Hinsi = {h| ∀t　h -> t, h ∈VN,
　　t ∈VT}

VN：非終端記号　　VT：終端記号

これらのカテゴリは一般に辞書項目に記載された品詞なっている。

従来のＢＵＰでは、予測したカテゴリがたとえ集合Hinsi に含まれていても、goal節を呼んでgoal節内部でdictionary節を呼んでいた。しかし、このようなgoalは、直接辞書引きを行うdictionaryの呼出しに置き換えることができる。

集合Hinsi の作成方法は、文法規則（辞書項目は除く）の左辺に一度も出て来ない文法カテゴリをHinsiの要素とすればよい。これも図1-1のトランスレータが自動的に検出し、goalの呼出しをdictionaryに置き換えることができる。

```
det(G,O,Info) --> [link(det,G)],
        goal(n,O),
        np(G,O,Info).
              ↓
det(G,O,Info) --> [link(det,G)],
        dictionary(n,O),
        np(G,O,Info).
```

図4-1 品詞の直接辞書引き

4.2トップダウンの成功部分木の登録

ＢＵＰは、解析時の再計算を避けるために一度成功や失敗したgoalは、成功ゴール、失敗ゴールとして登録している［松本83B ］。同様なことがトップダウンに於いても行なえる。ＤＣＧの呼出しの部分をnp(Arg)の形からtop(np,Arg)の形にし、top節の定義を［松本83B］のgoal節の定義とほぼ相似な図4-2のようにしておけばよい。

```
top(G,A,X,Y) :- (wf_top(G,X,_,_);
                 fail_top(G,X),!,fail),!,
                wf_top(G,X,A,Y).
top(G,A,X,Z) :- P=..[G,A,X,Z],call(P),
                assertz(wf_top(G,X,A,Z)).
top(G,A,X,Z) :- (wf_top(G,X,_,_);
                 assertz(fail_top(G,X))),!,fail.
```

図4-2 top節

5。結論

文法規則全体を、ボトムアップに解析を行なう規則とトップダウンに解析を行なう規則に自動的に分類し、それらの相互呼び出しを可能にすることにより、ボトムアップ法とトップダウン法を融合した新しいパーザの方式を提案した。また、文法カテゴリ中でいわゆる品詞（必ず終端記号となるカテゴリ）とそれ以外のカテゴリを区別するアルゴリズムを述べ、それらについて直接辞書引き行ない高速化をはかることができることを示した。

現在、我々の研究室で使用しているシステム（文法規則数約４００）に適用したところ、トップダウン規則とボトムアップ規則がほぼ半々になり、SUN-II Work-Station上のC-Prologインタープリタによる解析速度がボトムアップ法のみによるものと比べて３割程度向上した。

左外置の処理はボトムアップでも行なえるようになっているが、トップダウンによる解析法が有効であると考えている。今後は左外置の処理への応用を考えてみたい。［Pereira81］［今野84］

参考文献

［Pereira 80］：
Pereira, F. C. N. and Warren, D. H. D.,
"Definite Clause Grammar for Language Analysis -A Survey of the Formalism and a Comparison with Augmented Transition Networks",
Artif. Intell., 13,pp 231-278,May,1980

［Pereira 81］：
Pereira.F,
"Extraposition Grammar",
Am. J. Comput. Linguist., Vol.7, No.4,1981.

［松本83A ］：
松本裕治、田中穂積、平川秀樹、三吉秀夫、
安川秀樹、向井国昭、横井俊夫、
”Prologに埋め込まれたボトムアップパーサ：ＢＵＰ”、
Proc. of the Logic Programming Conference '83,
Mar.1983.

［松本83B］：
松本裕治、清野正樹、田中穂積、
”ＢＵＰの高速化”、
情報処理学会自然言語処理研究会資料39-7,1983

［今野84］：
今野聡、他、
”ボトムアップ構文解析における左外置の処理”、
日本ソフトウェア科学会第１回大会論文集、
pp223-226, 1984