自然言語処理　39－4
（1983. 9. 16）

Prologによる日本語文節DCG の生成

三吉秀夫　向井国昭　安川秀樹　平川秀樹　古川康一
（財団法人　新世代コンピュータ技術開発機構）

1．はじめに

現在ICOTでは本格的な自然言語処理システムのベースとなる日本語／英語の文法を開発中である。我々はすべての自然言語処理システムを１階述語論理を基礎としたProlog[Pereira 78] を用いてインプリメントし，構文解析のための文法記述も文脈自由文法（以後CFG と記す）に基くDCG(Definite Clause Grammar)モデルを採用する方針である。その理由として，現在人工知能向言語として注目されているPrologは自動バックトラック，ユニフィケーションを基本機能として持ち，Lispと同程度あるいはそれ以上の能力を持つ言語であるということ，また従来の言語では複雑な手続きで書かれていた処理がより簡潔に記述できる長所がある点があげられる。またDEC System-10 Prologに組込まれているDCG というパーザ[Pereira 80]は，CFG 規則がPrologプログラムの基本的シンタクスであるHorn節との類似性を利用して実装されているもので，文法カテゴリがPrologの述語として表現され，１つの文法規則がPrologの１つのHorn節にコンパイルされていると見ることができる。そのため文法カテゴリに引数を与えたり，文法カテゴリの間にPrologのプログラムを挿入できるため文脈依存的な情報も扱うことが可能である。また最近の言語学の分野では，従来変形を用いて分析を行っていた言語現象をCFG を用いて分析する文法理論も出現しており[郡司83,Gazdar 81 82]今後の自然言語処理の研究の流れとしてCFG を基礎とする文法を論理型プログラミング言語でインプリメントするのが有力と考えられるからである。

計算機による日本語処理を行う場合，活用の種類が多い，分かち書きの習慣がない，英語に比較して語順の自由度が大きい同音異義語が多い，省略が多い等日本語固有の特徴があるため[田中81]，工夫を要する。日本語の活用や助動詞の接続条件については国文法の分野でよく調べられているが，大規模な文法を作成する場合これらをそのままDCG に組込むのは保守性，拡張性の低下を招くので好ましくない。文法設計者は計算機内部の制御構造を意識しないで文法書の内容に近い形式で記述し，実際の処理プログラムへの変換はトランスレータやオプティマイザで行うのが望ましい。

このような観点から本稿では日本語文法開発の第一歩として，「行かせられたくなかったようだ」のような“動詞＋助動詞列”の文節の形態素解析を行うプログラムを中学校程度の文法書の知識から半自動生成する実験を行った結果を報告する。

2．プログラムの構成

本形態素解析プログラムは次の４つの部分から成る。

(1) 辞書項目
(2) 語尾処理述語
(3) 接続条件テーブル
(4) 基本文法

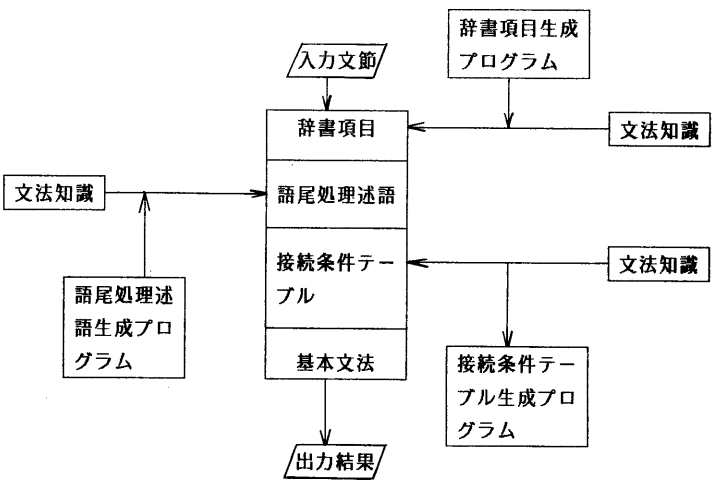

図 1. プログラムの構成

(1) －(3) はそれぞれ，辞書項目生成プログラム，語尾処理述語生成プログラム，接続条件テーブル生成プログラムにより生成される。これらの様子を図 1に示す。次節より(1) －(4) の項目について説明する。

2.1 辞書項目

本プログラムでは活用語である動詞，助動詞の辞書項目は活用しない語幹と活用語尾の２つの部分から成るというCFG 規則として記述される。

たとえば“学ぶ”という動詞の辞書項目は次の形式のDCG で記述される。

```
word(verb,manabu, K ,dan_5) --> gokan(mana),
  gobi(type(verb(dan_5,b)),K ).
```

この規則の左辺の第１引数verbは品詞が動詞であることを示す。第２引数manab は“学ぶ”の代表形（見出し）である。第３引数の変数は解析時に活用形がユニファイされる。第４引数dan _5 は五段活用であることを示す。右辺の第１要素gokan(mana) は“学ぶ”の語幹mana（学）を示す。右辺の第２要素gobiが語尾処理を司どる述語であり，その第１引数で示される活用の解析を行う。上の例ではバ行五段活用・動詞型の活用語尾の解析をし，第２引数の変数に活用形を返す。述語gobiについては 2.2 で説明する。動詞に関する辞書項目は図 2の辞書項目生成プログラムを「?- gen_w.」で起動することにより次々に生成される。

助動詞に関しては例外的な活用をするものが多いので辞書項目を直接記述する。ただし特定の活用型を持つものについては 2.2で述べる述語gobiを語尾処理に用いる。

次に助動詞の辞書項目をいくつか示す。

```
word(auxv,saseru,K,verb)  --> gokan(sa),
    gobi(type(verb(simo _1,s)),K).
```

これは使役の助動詞「させる」の辞書項目であり，左辺の第１引数auxvは品詞が助動詞であること，第２引数saseruは“させる”の代表形，第３引数の変数は解析時に活用形がユニファイされ，第４引数verbは活用型が動詞型であることを示す。右辺は動詞の辞書項目と同様で，語幹がsaで，サ行下一段・動詞型の活用語尾を持つことを示す。

```
word(auxv,nai ,K,adj) --> gokan(na),
      gobi(type(adj),K).
```

これは否定の助動詞「ない」の辞書項目で，語幹がna

```
prod_word((word(verb,Word,K,Kname)-->gokan(Stem),
        gobi(K,type(verb(Kname,Gname))))):-
  (Kname=dan_5,
     (Gname=k,Stem in [o,ka,ma,i];
      Gname=s,Stem in [o,ka,ke,wata];
      Gname=t,Stem in [u,ka,ma,mo];
      Gname=n,Stem in [si];
      Gname=m,Stem in [yo,ku,ka,su];
      Gname=r,Stem in [nobo,kuda,saga,ka,to,ya];
      Gname=w,Stem in [ka,omo,hiro,wara];
      Gname=g,Stem in [oyo,ko,huse,ao,nu];
      Gname=b,Stem in [koro,to,mana,hako]);
  Kname=kami_1,
     (Gname=a,Stem in ['','',o];
      Gname=k,Stem in ['',o,de];
      Gname=t,Stem in [o,ku,mi];
      Gname=n,Stem in ['',''];
      Gname=h,Stem in [''];
      Gname=m,Stem in [si,'',kaeri];
      Gname=r,Stem in [o,ka,ta];
      Gname=g,Stem in [su];
      Gname=z,Stem in [kan,to,ha];
      Gname=b,Stem in [a,o,no]);
  Kname=simo_1,
     (Gname=a,Stem in ['',tora,kanga];
      Gname=k,Stem in [ma,tasu,sa];
      Gname=s,Stem in [no,ki,mi];
      Gname=t,Stem in [su,a,soda];
      Gname=n,Stem in ['',kasa,tazu];
      Gname=h,Stem in [''];
      Gname=m,Stem in [arata,ho,to];
      Gname=r,Stem in [ku,ka,naga];
      Gname=g,Stem in [ko,na,ni];
      Gname=z,Stem in [ma,ha];
      Gname=d,Stem in [na,kana];
      Gname=b,Stem in [sira,ta,no]);
  Kname=kahen,
     (Gname=k,Stem in ['']);
  Kname=sahen,
     (Gname=s,Stem in [''];
      Gname=z,Stem in [an])          ),
  name(Gname,G1),name(Stem,S1),
  append(S1,G1,W),    name(Word,W).

gen_w :- prod_word(W),nl,write(W),
         expand_term(W,Y),assert(Y),fail.
```

図 2. 辞書項目生成プログラム

で，形容詞型の活用をすることを示す。

```
word(auxv,souda ,K,adjv) --> gokan(sou),
  gobi_1([(renyou,[d,e]) ,(syuusi,[d,a]) ],K).
```

これは伝聞の助動詞「そうだ」の辞書項目であり，語幹がsou で，形容動詞型の活用をすることを示している。gobi_1 という述語は，例外的な活用語尾処理をするためのもので，この例では，語尾が連用形のdeと終止形のdaしかないことを示す。

2.2 語尾処理述語

2.1で述べたように，語尾処理はgobiまたはgobi_

1 という述語によって行われ，解析時に第２引数に語尾の活用形を返す。

たとえばカ行五段・動詞型活用の語尾変化を処理するための述語gobiは次のように定義される。

```
gobi(type(verb(dan_5,k)),mizen) --> [k,a].
gobi(type(verb(dan_5,k)),renyou)  --> [k,i].
gobi(type(verb(dan_5,k)),syuusi)  --> [k,u].
gobi(type(verb(dan_5,k)),rentai)  --> [k,u].
gobi(type(verb(dan_5,k)),katei )  --> [k,e].
```

述語gobiは動詞型，形容詞型，形容動詞型の活用の語尾処理を行い，これらは図 3の語尾処理述語生成プログラムを「?- gen_r.」で起動することによって生成される。

2.3 接続条件

本プログラムでは隣り合う２つの単語の間に接続が可能かどうかチェックするための述語を文法規則中に外部条件として組込んでいる。文法書によれば，助動詞に関

```
inflection(dan_5, [a,i,u,u,e,e],
        [k,s,t,n,m,r,w,g,b]).
inflection(kami_1,[i,i,iru,iru,ire,[iro,iyo]],
        [a,k,t,n,h,m,r,g,z,b]        ).
inflection(simo_1,[e,e,eru,eru,ere,[ero,eyo]],
        [a,k,s,t,n,h,m,r,g,z,d,b]).
inflection(kahen, [o,i,uru,uru,ure,oi],
        [k]                    ).
inflection(sahen, [[i,e,a],i,uru,uru,ure,[iro,eyo]],
        [s,z]).

form(mizen,[X|_],Y):-in_1(Y,X).
form(renyou,[_,X|_],Y):-in_1(Y,X).
form(syuusi,[_,_,X|_],Y):-in_1(Y,X).
form(rentai,[_,_,_,X|_],Y):-in_1(Y,X).
form(katei,[_,_,_,_,X|_],Y):-in_1(Y,X).
form(meirei,[_,_,_,_,_,X],Y):-in_1(Y,X).

in_1(M,M):-atomic(M),!.
in_1(M,N):-in(M,N).

prod((gobi(type(verb(X,Y)),Z)-->U)):-inflection(X,G,Gyou),
        Y in Gyou,form(Z,G,W),resolve_atom(W,NW),
                                (Y\==a,append([Y],NW,U);
                                Y==a,U=NW).

prod((gobi(type(verb(dan_5,X)),renyou)-->U)):-(X,U) in
        [(k,[i]),(t,[t]),(n,[n]),(m,[n]),
         (r,[t]),(w,[t]),(g,[i]),(b,[n])].

prod((gobi(type(adj),K)-->U)):-(K,U) in
        [(mizen,[k,a,r,o]),
        (renyou,[k,a,t]),(renyou,[k,u]),
        (syuusi,[i]),
        (rentai,[i]),
        (katei, [k,e,r,e])].

prod((gobi(type(adjv),K)-->U)):-
        (K,U) in
        [(mizen,[d,a,r,o]),
        (renyou,[d,a,t]),(renyou,[d,e]),(renyou,[n,i]),
        (syuusi,[d,a]),
        (rentai,[n,a]),
        (katei,[n,a,r,a])].

gen_r :- prod(X), /* ttynl,display(X),  */
         expand_term(X,Y),fout(Y),fail.
```

図 3. 語尾処理述語生成プログラム

する接続条件は次の５種類統語的な要素によって記述されている。

(i)　品詞（動詞，助動詞，助詞，用言，etc ）
(ii)　活用の種類（五段，上一段，下一段，etc)
(iii) 活用形（未然形，連用形，終止形，etc ）
(iv)　活用型（動詞型，形容動詞型，etc ）
(v)　特定の単語，あるいは特定の単語以外

そこで入力文節を解析中に助動詞が見つかった場合，本プログラムではそれに前接する単語との接続可能性をチェックを次の形式の'link'という述語を用いて行う。

```
link(Auxverb,[Hinsi,
              Katuyou _syurui,
              Katuyou _kei,
              Katuyou _type,
              Individual_word]).
```

これは助動詞の接続条件テーブルであり，基本的にはPrologのfactとして与えられる。第１引数の助動詞が第２引数の統語的要素を持つ単語に接続可能であることを示す。linkの呼び出し側は 2.4で述べる基本文法中にPrologの外部条件として組込まれている。

第１引数Auxverb は各助動詞の代表形で，

Auxverb ∈ {reru,rareru,masu,　..... }

である。

第２引数は上記５種類の統語的要素のリストで，

Hinsi ∈ {meisi,dousi,keiyoudousi,
　　yougen,katuyougo, ... }
Katuyou _syurui　∈　{dan_5,kami_1,
　　simo_1,kahen,sahen }
Katuyou _kei ∈　{mizen,renyou,syuusi,
　　rentai,meirei }
Katuyou _type　∈　{verb,adj,tokusyu}
Individual_word　∈　単語の代表形の集合

である。

次に接続テーブルの例をいくつか示す。助動詞「れる」に対する接続条件「五段・サ変の未然形につく」は次のように表される。

```
link(reru,[dousi,dan_5,mizen,KT,WD]).
link(reru,[dousi,sahen,mizen,KT,WD] ).
```

「～ 以外の･･･ につく」という接続条件の記述に対してはPrologのnot とmemberを用いて表す。たとえば過去・完了の助動詞「た」の接続条件「ぬ，う，よう，まい以外の助動詞の連用形につく」は次のように表現される。

```
link(ta,[jodoushi,KS,renyou,KT,WD)　:
    not(member(WD,[nu ,u,you,mai])).
```

このlinkという接続条件テーブルについてもユーザが直接入力するのではなくて，文法書で述べられている表現にできるだけ近い形をrow dataとして入力し，接続条件テーブルへの変換は接続条件テーブル生成プログラムが行う。

raw dataは次の形式で記述する。

```
aux _con _table （助動詞の代表形，
      [＜接続条件リスト－１＞，
       ＜接続条件リスト－２＞，
       ・・・・・・・・・
       ＜接続条件リスト－ｎ＞] ）.
```

第２引数の要素である＜接続条件リスト－ｍ＞は前記５種類の統語的要素のうちで文法書に陽に記述されているものについて ,"hinshi","katuyou _syurui","katuyou_kei","katuyou _type","indiv" というファンクタの引数にしたタームのリストである。

たとえば助動詞「ない」に関する接続条件の記述，

“動詞および助動詞「れる・られる」，「せる・させる」，「たがる」の未然形につく。五段動詞「ある」にはつかない。助動詞「たい」「そうだ」「だ」「ようだ」「みたいだ」「らしい」，形容詞，形容動詞の連用形につく。”

に対するraw dataをユーザは次のように入力する。

```
aux _conn_table(nai, [
  [hinsi(dousi),katuyou _kei(mizen),
    indiv(except(aru))],
  [hinsi(dousi),katuyou _kei(mizen),
    indiv(reru,rareru,seru,saseru,tagaru)],
  [hinsi(dousi),katuyou _kei(renyou),
    indiv(tai,souda,da,youda,mitaida,rasii)],
  [hinsi(keiyousi,keiyoudousi ),
    katuyou _kei(renyou)]　　] ).
```

これを接続条件生成プログラムにかけることにより，図 4の形式の「ない」に関する接続条件テーブルが得られる。

テーブル中で統語的要素に変数が与えられているものはdon't careであることを示す。たとえば品詞が動詞であるものの活用型は動詞型であることが決まっているので活用型を指定する必要はない。

```
link(nai,[dousi,KS,mizen,KT,WD]):-
      not(member(WD,[aru])).
link(nai,[jodousi,KS,mizen,KT,reru]).
link(nai,[jodousi,KS,renyou,KT,tai]).
link(nai,[keiyousi,KS,renyou,KT,WD]).
link(nai,[keiyoudousi,KS,renyou,KT,WD]).
link(nai,[jodousi,KS,renyou,KT,souda]).
link(nai,[jodousi,KS,renyou,KT,da]).
link(nai,[jodousi,KS,renyou,KT,youda]).
link(nai,[jodousi,KS,renyou,KT,mitaida]).
link(nai,[jodousi,KS,renyou,KT,rasii]).
link(nai,[jodousi,KS,mizen,KT,rareru]).
link(nai,[jodousi,KS,mizen,KT,seru]).
link(nai,[jodousi,KS,mizen,KT,saseru]).
link(nai,[jodousi,KS,mizen,KT,tagaru]).
```

図 4. 「ない」の接続条件テーブル

2.4 基本文法

このプログラムで用いられている日本語文節を生成する文法規則を図 5に示す。この文法の意味は，

・文節は唯１個の動詞あるいは唯１個の動詞と助動詞列から成る（ (1),(2) ）。

・助動詞列は唯１個の助動詞あるいは唯１個の助動詞と助動詞列からなる（ (3),(4) ）。

なお，(5),(6) は辞書引き時に統語要素のリストを作るための付加的な規則である。

文節のカテゴリに相当する'v' の第１引数には，解析後に文節を構成する単語の代表形のリストがアサインされる。 (3), (4) の規則の右辺で，{ } で囲まれている述語linkが 2.3で述べた助動詞の接続条件をチェックする付加手続きである。aux _verbを実行して助動詞が１個みつかった後，左辺の第３引数に直前の単語の統

```
(1) v(X,Y) --> verb(X,Y,Verb_Type).

(2) v([A|C],[B|D]) --> verb([A],[B],Verb_Type),auxv(C,D,Verb_Type).

(3) auxv([X],[Y],Precedent_Type) -->
       aux_verb([X],[Y],Aux_Type),{link(X,Precedent_Type)}.

(4) auxv([X|X1],[Y|Y1],Precedent_Type) --> aux_verb([X],[Y],Aux_Type),
                    {link(X,Precedent_Type)},auxv(X1,Y1,Aux_Type).

(5) verb([Daihyou_kei],[Katsuyou_kei],[doushi,
                                       Katsuyou_shurui,
                                       Katsuyou_kei,
                                       verb,
                                       Daihyou_kei]) -->
        word(verb,Daihyou_kei,Katsuyou_kei,Katsuyou_shurui).

(6) aux_verb([Daihyou_kei],[Katsuyou_kei],[jodoushi,
                                           _,
                                           Katsuyou_kei,
                                           Type,
                                           Daihyou_kei]) -->
        word(auxv,Daihyou_kei,Katsuyou_kei,Type).
```

図 5. 基本文法

語要素のリストが与えられているので，2つの単語の接続可能性をlinkでチェックしている。つぎにこのプログラムで形態素解析を行った実行例を図 6に示す。

3．まとめ

本実験により，中学校の文法書に書かれている程度の文法的制約事項を満たすように形態素処理を行うプログラムが一種のメタルールを用いることにより，比較的ユーザに分り易い形式を用いて記述，生成できることがわかった。大規模な自然言語処理システムを構築することを考えるた場合，エンドユーザは言語学者のような計算機が専門でない人も含まれるので，エンドユーザが文法的な知識を直接計算機処理に向いた形式で記述するのではなく，文法的な知識はエンドユーザ記述言語で記述し，オブジェクトプログラムへの変換またそれに伴う最適化の手法の導入などは専用のトランスレータ，オプティマイザが行うという形態をとるのが妥当であろう。この観点から，本実験で用いた手法がその手掛りになると考えられる。

現在ICOTで開発中の高機能構文解析システムは，エンドユーザ言語によるメタな文法記述からDCG,BUP へトランスレートする方式を検討中である。エンドユーザ記述言語については研究が始ったところであり，その仕様については今後の検討を待たねばならないが,.知識表現言語で用いられているようなクラス・マルチプルインヘリタンスの実現，あるいは高度のマクロ機能の導入などを考えている。

本形態素解析プログラムはDEC2060 上で稼働するエジンバラ版Prologですべてインプリメントされている。今後はエンドユーザ記述言語，及びトータルな日本語文法を開発し，意味理解に重点をおいた日本語理解システムの開発を行う予定である。

```
Enter the Sentence
|: manabu.

Tango = [manabu]
Katsuyou = [syuusi]
Runtime = 86 msec


Enter the Sentence
|: manabimasu.

Tango = [manabu,masu]
Katsuyou = [renyou,syuusi]
Runtime = 392 msec


Enter the Sentence
|: manabumasu.

no


Enter the Sentence
|: manabaseraretakunakattayouda.

Tango = [manabu,seru,rareru,tai,nai,ta,youda]
Katsuyou = [mizen,mizen,renyou,renyou,renyou,rentai,syuusi]
Runtime = 1295 msec
```

図 6. 実行例

［謝辞］

本研究の機会を与えて頂いた渕一博ICOT研究所長，有益な御討論をして頂いた第２研究室の皆様，WG3 の諸先生方に感謝致します。

［参考文献］

1. 田中穂積，その他：自然言語処理技術と言語理論，電子技術総合研究所調査報告 205号，1981
2. Pereira,L., Pereira,F.,and Warren,D. : User's guide to DECsystem-10 Prolog,Div.de Informatica, LNEC,Lisbon and Dept.of AI,University of Edinburgh ,1978
3. Pereira,F. and Warren,D.：Definite Clause Grammar for Language Analysis - A survey of the Formalism and a Comparison with Augumented Transition Networks, Artificial Intelligence, 13, pp231-278, 1980.
4. 郡司隆男：日本語におけるコントロール，情報処理学会，自然言語処理研究会35-3，1983
5. Gazdar,G.：Unbounded dependencies and coordinate structure, Linguistic Inquiry 12, pp155-184. 1981
6. Gazdar,G. and Geoffrey,K.P. ：Generalized phrase structure grammar:a theoretical synopsis, Indiana Univ. Linguistic Club, Bloomington, Indiana , 1982
7. 橋本武：やさしい文法，学燈社，1980
8. 寺瀬光男：中学口語文法，梧桐書院，1976
9. 松沢利彦：拡張LINGOL上での用言の形態素処理に関する研究，情報処理学会自然言語処理研究会26-5，1981
10. 松本裕治，田中穂積：Prologに埋め込まれたbottom-up parser:BUP，情報処理学会自然言語処理研究会34-6 1982