〔研究ノート〕

ドイツ語の機械解析に向けて (1)[1)]

山田善久

0. 序

言語理論に関心をもつ者にとって，理論をシミュレートさせることによって，理論の妥当性を検証したり，あるいは新たな問題発見のてがかりとして，コンピュータは魅力的なツールである。とくに近年のいわゆるパソコンの急激な技術革新と普及により，文科系の研究者の間でもコンピュータはいっそう身近なものとなりつつある。

コンピュータ工学の分野では，最近の AI（人工知能）ブームの高まりの中で，自然言語処理ないしは自然言語理解といった分野が重点的な研究課題の一つとなり，例えばその応用の成果の一つである機械翻訳システムが，まだまだ不十分なものとはいえ，企業サイドで続々開発・商用化されるに及んでいる[2)]。

「コンピュータ言語学」という学問分野が応用言語学の特殊な一領域の枠

を超えて広く市民権を得，熱い眼差しをもってさえ語られるようになったのもこうした気運と無縁ではないだろう。

従来国内で重点的に研究されてきた言語は，社会的要請からも当然のことであるが，英語と日本語である。筆者は一ゲルマニストとして，ドイツ語の機械処理に関心をもち，現在パソコン上の Prolog 処理系を用いたドイツ文解析システムを作成中である[3]が，ここでその一端を紹介するとともに，とくにドイツ語を処理する場合の問題点に言及しておきたい。

1. 自然言語処理における文解析のプロセス

自然言語処理における文解析のプロセスは，一般に次の四つのステップを踏むとされる[4]。

Ⅰ．形態素解析
Ⅱ．構文解析
Ⅲ．意味解析
Ⅳ．文脈解析（談話解析）

以上の流れは，文法理論における形態論，統語論，意味論，テキスト文法（ないしは語用論）と，いわばパラレルな関係にある。Ⅰでは，与えられた文連鎖を単語に分け，必要ならば形態素規則によって原形を取り出し，辞書引き可能な状態にする。Ⅱでは，辞書と構文の規則によって文の構造記述を与える。Ⅲでは，例えば格関係といった文の意味情報を与えたり，文の整合性や同義性を判断させる。Ⅳでは，代名詞の照応関係とか前提など，文を超えたレベルの情報を扱う。このうちⅢおよびⅣは，理論的にも，とくにその定式化については不明の点が多く，本格的な解明は今後の課題であり，現状の実用システムにおいても，当面の処理に必要で，比較的定式化しやすい現象を経験的手法によって組み込んでいるのが現状のようである。筆者のシステムも目下のところは，選択制限ないしは分布と呼ばれる文の整合

性を判定させるための意味的情報[5]は扱っているものの，ほぼIIまでの処理に限定している。以下本号では主としてIの形態素解析について検討し，文解析のメインをなすIIの構文解析については次号で稿をあらためて考察することにしたい。

2. ドイツ語の形態素解析

前章で述べたとおり，形態素解析とは，与えられた文連鎖を単語に分け，形態素規則によって原形情報を得，辞書引き可能な状態にする一連の作業であるが，筆者が対象としているドイツ語など欧米語では，（文字言語を前提とする限り）あらかじめ単語の分かち書きがなされているので，文連鎖を単語に切り分ける作業は省略できる。一方，とくにドイツ語のように語形変化の激しい言語では，形態素規則の導入は不可欠であり，その規則の数もある程度膨大なものとならざるを得ない。以下ではドイツ語の形態素規則をPrologで表現する手法を，人称変化詞（動詞類）および格変化詞（名詞類）の二つに分けて考えてみることにする。

2.1. 人称変化詞（動詞類）の形態素規則

ドイツ語の形態素規則で問題となるのは，もっぱら語尾の変化である。例を動詞にとって言うと，er singt における singt という定形から singen という不定形を得る手順である。この処理のためには，文字列の操作，すなわち上の例に即していうと singt を sing と t に分解し，語幹 sing に不定形語尾 en を付加する処理が必要となる。次の Prolog によるプログラムは，この処理をおこなう動詞現在人称変化の規則である。

```
v(FV, Typ, Pf, sg, 1, ps, Syn, Sem, IV, Jpn) :-
        name(FV, FCode),
```

```
        append(SCode, [101], FCode),
        append(SCode, [101, 110], ICode),
        name(IV, ICode),
        v(IV, inf, Typ, Pf, Syn, Sem, Jpn).          %１人称単数現在
v(FV, Typ, Pf, sg, 2, ps, Syn, Sem, IV, Jpn) :-
        name(FV, FCode),
        append(SCode, [115, 116], FCode),
        append(SCode, [101, 110], ICode),
        name(IV, ICode),
        v(IV, inf, Typ, Pf, Syn, Sem, Jpn).          %２人称単数現在
v(FV, Typ, Pf, sg, 3, ps, Syn, Sem, IV, Jpn) :-
        name(FV, FCode),
        append(SCode, [116], FCode),
        append(SCode, [101, 110], ICode),
        name(IV, ICode),
        v(IV, inf, Typ, Pf, Syn, Sem, Jpn).          %３人称単数現在
v(V, Typ, Pf, pl, 1, ps, Syn, Sem, V, Jpn) :-
        v(V, inf, Typ, Pf, Syn, Sem, Jpn).           %１人称複数現在
v(FV, Typ, Pf, pl, 2, ps, Syn, Sem, IV, Jpn) :-
        name(FV, FCode),
        append(SCode, [116], FCode),
        append(SCode, [101, 110], ICode),
        name(IV, ICode),
        v(IV, inf, Typ, Pf, Syn, Sem, Jpn).          %２人称複数現在
v(V, Typ, Pf, pl, 3, ps, Syn, Sem, V, Jpn) :-
        v(V, inf, Typ, Pf, Syn, Sem, Jpn).           %３人称複数現在
```

標準 Prolog では，アトムと呼ばれる定数項（言語になぞらえて言えば，文の最小単位である単語にあたる）が操作の最小単位であり，例えば LISP における EXPLODE ないしは UNPACK のような，アトムを直接文字キャラクタに分解する関数にあたるものが用意されていないので，文字列の処理は幾分複雑なものになる。上のプログラムでは，アトムを文字コードのリストに変換する組み込み述語 name/2 およびリストを連結する述語 append/3 によって，この処理を実現している。

ここで上のプログラムの意味するところを少し細部に立ち入って解説してみたいが，その前に Prolog になじみのない読者のために，Prolog のプログラムの形式と動作原理を簡単に紹介しておくことにする[6]。Prolog のプログラムは，事実を記述するユニットの集合，つまり一種のデータベースとみなすことができる。プログラムの実行は，このデータベースに対する質問と，Prolog インタプリタ側のデータベース探索結果の解答という原理によっておこなわれる。事実を記述する個々のユニットは節と呼ばれ，そのプログラムの形式は，次の二つの一般式にまとめることができる。

$P(T_0, \cdots, T_n).$ $\langle n \geqq 0 \rangle$

$P(T_0, \cdots, T_n) :- Q_1(T_0, \cdots, T_n), \cdots, Q_m(T_0, \cdots, T_n).$ $\langle m \geqq 1, n \geqq 0 \rangle$

上の式で，P, Q_m はそれぞれ述語と呼ばれ，（ ）の中の T_n は引数または項と呼ばれる。この引数はなくてもよいし，また任意個許容することができる。節の末尾にはピリオドが置かれる。最初の節は，述語 P/n（スラッシュの後の数字は引数の数を示す）が，無条件で事実であることを表わす。筆者のシステムでは，辞書部門の語彙項目の記載にこの形式が用いられる。第2の節は少し複雑な形をしているが，中ほどにある :- という記号が，自然言語の if にあたる意味を表わし，この右辺（節の本体 body と呼ばれる）が成立する場合に限り，左辺（節の頭部 head）が事実となることを表わ

す。つまり一種の条件節になっているわけである。本稿の形態素規則をはじめ一般に文法規則はすべてこの型によって記述される。なおこの場合，右辺の本体には任意個の述語を置くことが可能であるが，左辺の頭部は必ず1個の述語でなければならない。

さて先ほどのプログラムに戻って，具体的にそのプロセスを追っていこう。各節の頭部（左辺）の述語 v/10 は，定形動詞の各変化形を表現するもので，第1引数が語形，第3引数から第5引数までがそれぞれ数，人称，時制を表わす。例として，プログラムの最初の節（1行目から6行目）を取り上げると，第3引数から第5引数までが，sg, 1, ps と定数[7]になっており，これで単数1人称現在であることが記述されている。その他の引数の意味は，ここでの処理には当面必要ないので今はふれない。次に条件を記述する部分である本体（右辺）の方を見ていこう。まず最初の述語 name/2 の第1引数が変数 FV となっているが，これは頭部の第1引数と同じである。この変数に定形動詞の候補である何らかの語形がユニファイ（パターン・マッチ）[8]することになるが，これが述語 name/2 によって文字（キャラクタ）コードのリストに変換され，第2引数 FCode にその値が返される。例えば，singe という語形の場合だと，そのキャラクタ・コードのリストである［115, 105, 110, 103, 101］（“singe”）という値が返されることになる。次に2番目の述語 append/3 の第3引数が FCode であり，上のキャラクタ・コードのリストはこれとユニファイする。この述語 append/3 は，第1引数と第2引数のリストを連結し，連結したリストを第3引数とする述語[9]であるが，ここではこれを逆向きに使っている。すなわち，ここでは第1引数が変数，第2引数および第3引数が定数であり，第3引数のリスト（ここでは “singe”）の末尾から第2引数のリスト［101］（“e”）を取り去ったものが第1引数 SCode である，と読むことができる。これは言語の文法記述に即して言うと，singe から語尾 e を取って語幹（すなわち sing）を求める操作である。次に本体の第3番目の述語 append/3 では，先に SCode とユニファ

イしていた語幹 “sing” が同じくこの第1引数である SCode とユニファイし，これを第2引数 [101, 110]（“en”）と連結し，この連結されたリスト（“singen”）が第3引数 ICode に入る。つまり不定形を求める操作がここでおこなわれる。次の第4番目の述語 name/2 は，キャラクタ・コードのリストをアトムに戻すためのもので，上で append/3 の第3引数 ICode に値として入っていたキャラクタ・コードのリスト “singen”（[115, 105, 110, 103, 101, 110]）が，この述語の第2引数 ICode とユニファイし，第1引数 IV にそのアトム singen が返される。第5番目の述語 v/7 は，不定形の語彙項目であり，例えば singen は，辞書部門のデータベースに次のように記載されている。

v(singen, inf, i, h, _, _, [[], 歌, う, わない, った]).

先の述語 v/7 の第1引数 IV にはアトム singen が入っており，これが上の語彙項目とユニファイすることにより，この節全体が成功する。つまり，singe は，不定形が singen であり，その1人称単数現在形である，という情報が得られたことになる。プログラムの7行目以下の節も同様にして，単数2人称，3人称，複数1，2，3人称の各語尾変化を規則化したものである。

以上はいわゆる規則変化の動詞の場合であるが，ドイツ語にはさらに，sein, haben, werden や語幹の母音が交替する fahren（er fährt），sprechen（er spricht）のような一連の不規則変化の動詞がある。これらの動詞の変化のパターンは，各語彙に特有のものであり，上の規則変化の動詞の場合のように形態素規則を導入することはできないので，その個々について独立した語彙項目を設ける必要がある。例えば，sein の各人称変化形（定形）の辞書記述は，先の定形動詞のところで導入したのと同じ述語 v/10 を用いて，次のようになる。

```
v(bin, il, s, sg, 1, ps, fr, [], sein, [[], で, ある, [], あった]).
v(bist, il, s, sg, 2, ps, fr, [], sein, [[], で, ある, [], あった]).
v(ist, il, s, sg, 3, ps, fr, [], sein, [[], で, ある, [], あった]).
v(sind, il, s, pl, 1, ps, fr, [], sein, [[], で, ある, [], あった]).
v(seid, il, s, pl, 2, ps, fr, [], sein, [[], で, ある, [], あった]).
v(sind, il, s, pl, 3, ps, fr, [], sein, [[], で, ある, [], あった]).
```

過去形および過去分詞形の記述についても，以上の現在形の場合と同様の原則があてはまる。すなわち過去人称変化および規則動詞の過去基本形，過去分詞形については，語尾のパターンを処理する形態素規則が導入され，不規則動詞の過去基本形，過去分詞形については，別途語彙項目を設けて処理することになる。過去人称変化の規則は，手順としては現在人称変化の規則と同様であり，多少煩雑になるが，以下にこの規則もあげておこう。

```
v(FV, Typ, Pf, sg, 1, pt, Syn, Sem, IV, Jpn) :-
    v(FV, pt, Typ, Pf, Syn, Sem, IV, Jpn).          %1人称単数過去
v(FV, Typ, Pf, sg, 2, pt, Syn, Sem, IV, Jpn) :-
    name(FV, FCode),
    append(SCode, [115, 116], FCode),
    name(PV, SCode),
    v(PV, pt, Typ, Pf, Syn, Sem, IV, Jpn).          %2人称単数過去
v(FV, Typ, Pf, sg, 3, pt, Syn, Sem, IV, Jpn) :-
    v(FV, pt, Typ, Pf, Syn, Sem, IV, Jpn).          %3人称単数過去
v(FV, Typ, Pf, pl, 1, pt, Syn, Sem, IV, Jpn) :-
    name(FV, FCode),
    (append(SCode, [101, 110], FCode)
    ; append(SCode, [110], FCode)
```

```
    ),
    name(PV, SCode),
    v(PV, pt, Typ, Pf, Syn, Sem, IV, Jpn).          %1 人称複数過去
v(FV, Typ, Pf, pl, 2, pt, Syn, Sem, IV, Jpn) :-
    name(FV, FCode),
    append(SCode, [116], FCode),
    name(PV, SCode),
    v(PV, pt, Typ, Pf, Syn, Sem, IV, Jpn).          %2 人称複数過去
v(FV, Typ, Pf, pl, 3, pt, Syn, Sem, IV, Jpn) :-
    name(FV, FCode),
    (append(SCode, [101, 110], FCode)
    ; append(SCode, [110], FCode)
    ),
    name(PV, SCode),
    v(PV, pt, Typ, Pf, Syn, Sem, IV, Jpn).          %3 人称複数過去
```

上の規則ではそれぞれ，過去人称変化語尾を取った語形を，それぞれの節の最後にある述語 v/8 とユニファイさせる手続きが記述されている。この v/8 という述語は，過去基本形を表わすもので，規則動詞の場合は，別に次のような形態素規則によって基本形の処理がなされる。ここでは，語尾 te を取って en をつけた語形を辞書の不定形の記載項目とユニファイさせる手続きが記述されている。

```
v(PV, pt, Typ, Pf, Syn, Sem, IV, Jpn) :-
    name(PV, PCode),
    append(SCode, [116, 101], PCode),
    append(SCode, [101, 110], ICode),
```

```
    name(IV, ICode),
    v(IV, inf, Typ, Pf, Syn, Sem, Jpn).          %規則動詞過去基本形
```

次は過去分詞形を処理する形態素規則である。同様に，前綴り ge および語尾 t を取ったあと en をつけた語形を不定形の辞書項目とユニファイさせる手続きが記述されている。前綴りおよび語尾の二つの接辞の処理を 2 ステップでおこなっているので，述語 append/3 が三つも並ぶことになり，多少煩雑な処理となっている。

```
v_pp(PP, pp, Pf, Typ, Syn, Sem, IV, Jpn) :-
    name(PP, PfCode),
    append([103, 101], SCode1, PfCode),
    append(SCode0, [116], SCode1),
    append(SCode0, [101, 110], ICode),
    name(IV, ICode),
    v(IV, inf, Typ, Pf, Syn, Sem, Jpn).          %規則動詞過去分詞形
```

いわゆる過去現在動詞（Präteritopräsens）と呼ばれる話法の助動詞および動詞 wissen の人称変化形についても，上の過去人称変化の形態素規則で処理することができる。ただこのクラスの動詞は，単数定形の幹母音が不定形と異なるので，これについては先の不規則動詞の場合と同様，別に語彙項目をたてておく必要がある。

動詞類の形態素規則の最後に，分離動詞についてふれておこう。この場合は，文中で不連続になっている定形動詞と前綴りを再結合して，辞書引きをおこなわせるわけであるが，文という単位が解析の前提となることから，次のプロセスである構文解析に立ち入ることになる。構文解析のプロセスでは，動詞の位置のヴァリアント（前置，中置，後置）を適切に処理するた

め，解析すべき表層文の定形動詞を文末に移動する規則を設け，副文における SOV 語順を基底構造として処理している。これにより，複合時称や助動詞構文など述語が 2 語以上からなる場合でも，連続した句の並びとして処理することができる。この述語句を処理するために，述語 verb1/11 が導入される。分離動詞もこの一環として処理される。次は verb1/11 を定義したプログラムの一部である。3 行目以下が分離動詞を記述する部分である。

```
verb1(A, B, Typ, Num, Psn, Tns, Syn, Sem, IV, Tree, Jpn) :-
    v(A, B, Typ, Num, Psn, Tns, Syn, Sem, IV, Tree, Jpn).
verb1([Prx, FV|X], X, Typ, Num, Psn, Tns, Syn, Sem, _, [v, TV],
Jpn) :-
    v(FV, Typ, Pf, Num, Psn, Tns, Syn1, Sem1, IV, Jpn1),
    name(Prx, PrCode),
    name(IV, ICode),
    append(PrCode, ICode, TVCode),
    name(TV, TVCode),
    v(TV, inf, Typ, Pf, Syn, Sem, Jpn).
```

述語 verb1/11 の第 1 引数および第 2 引数は差分リスト（d-list）[10]である。例として，

Er kommt heute in Berlin an.

という文を取り上げて，プログラムの動きを見ていこう。上で述べたように，表層文の定形動詞は，文末に移動して処理され，上の文は，次のような Prolog のリストの形で，以降の解析プロセスに引き継がれる。

[er, heute, in, Berlin, an, kommt]

ここで，[er, heute, in, Berlin] までが解析され，続く [an, kommt] を解析する過程で，上記述語 verb1/11 とのユニフィケーションが実行される。1行目から2行目にかけての節は，単純動詞の場合を記述するもので，ここでユニフィケーションが失敗し，3行目以下の節にバックトラックがおこる。この節では，差分リスト内の変数 Prx および FV に，an, kommt がそれぞれマッチする。以降の流れは他の形態素規則と同様，name/2 と append/3 を使ったもので詳述は省くが，変数 Prx とマッチしている an と，変数 FV とマッチしている kommt の不定形 kommen を結合した結果（ankommen）を，この節の本体の最後の述語 v/7 とユニファイさせることにより，辞書引きをおこなう過程が記述されている。この場合，辞書部門には，単純動詞の不定形と同様に，分離動詞 ankommen は次のように記載されている。

v(ankommen, inf, i, s, fr, dir, [[], 到着, する, し, した]).

なお分離動詞の場合，過去分詞および zu 不定詞において，ankommen → angekommen, anzukommen のように一語の形をとるが，次のプログラムはそれぞれこの処理をおこなうものである。

[過去分詞]

```
v_pp(TP, pp, Pf, Typ, Syn, Sem, TV, Jpn) :-
        name(TP, TPCode),
        append(PxCode, [103, 101|SCode], TPCode),
        PxCode ¥== [],
        name(PP, [103, 101|SCode]),
```

```
    v_pp(PP, pp, Pf0, Typ0, Syn0, Sem0, IV, Jpn0),
    name(IV, ICode),
    append(PxCode, ICode, TVCode),
    name(TV, TVCode),
    v(TV, inf, Typ, Pf, Syn, Sem, Jpn).
```

過去分詞の場合は，まず接辞 ge をポイントに分離前綴りと基礎動詞の過去分詞に分け，この過去分詞の不定形に分離前綴りをつけた形（分離動詞の不定形）を，単純動詞の場合と同様，節の最後の述語 v/7 によって辞書引きさせる。

```
[zu 不定詞]
  v([V|X], X, Typ, Num, Psn, Tns, Syn, Sem, IV, [v, V], Jpn) :-
    v(V, Typ, Pf, Num, Psn, Tns, Syn, Sem, IV, Jpn).
  v([zu, IV|X], X, Typ, Num, Psn, ps, Syn, Sem, IV, [v, [zu, IV]],
  Jpn) :-
    v(IV, inf, Typ, Pf, Syn, Sem, Jpn).
  v([ZV|X], X, Typ, Num, Psn, ps, Syn, Sem, TV, [v, ZV], Jpn) :-
    name(ZV, ZVCode),
    append(PxCode, [122, 117|ICode], ZVCode),
    append(PxCode, ICode, TVCode),
    M ¥== [],
    name(TV, TVCode),
    v(TV, inf, Typ, Pf, Syn, Sem, Jpn).
```

zu 不定詞は，定形動詞と同じく，差分リストを用いた述語 v/12 の中で記述される。この述語 v/12 は，トップダウンによる構文解析の最終局面に

あたり，本来構文解析部門に属するものである。さて上のプログラムで，述語 v/12 を定義する三つの節のうち，最初の節は定形動詞，第二の節は単純動詞の zu 不定詞の定義であり，最後の第三の節（6行目から12行目まで）が，分離動詞の zu 不定詞の定義となっている。処理の流れは先の過去分詞の場合と同様であり，例えば anzukommen を例にとると，接辞 zu を取った結果の ankommen という形を，最後の述語 v/7 によって，辞書の不定詞の語彙項目とユニファイさせる。この場合は，先述の ankommen の語彙項目とユニファイすることになる。

2.2. 格変化詞（名詞類）の形態素規則

ここでは冠詞類，名詞，代名詞，形容詞の格変化規則を扱う。この場合も，先の動詞の形態素規則と同様，語尾の操作が中心となるので，プログラムの考え方自体も同じである。以下順を追って見ていくことにする。

まず冠詞類であるが，この場合は語彙が限られているので形態素規則を導入せずに，個々の形態を辞書に登録しておくことも可能である。しかし，例えば不定冠詞と同じ変化の mein, dein, sein, ... 等の個々の変化形をいちいち辞書に列挙するのは，スマートな処理とは言い難いし，メモリの効率の点でも得策ではない。筆者の解析システムでは，定冠詞と不定冠詞については個個の変化形を辞書項目とし，いわゆる定冠詞類および不定冠詞類については形態素規則を導入するという方法をとっている。次のプログラムは定冠詞および不定冠詞の辞書記述である。それぞれ述語 det/5 で表わされ，第1引数が語形，第2引数が定・不定を示す素性，第3引数が性，第4引数が格，第5引数が日本語訳を示す。この日本語訳は，後の構文解析の項でふれることになるが，本システムでは冠詞を訳出しないので，ヌルリストとしている。

```
[定冠詞]
    det(der, def, m, nom, []).
```

```
det(des, def, m, gen, []).
det(dem, def, m, dat, []).
det(den, def, m, akk, []).
det(die, def, f, nom, []).
det(der, def, f, gen, []).
det(der, def, f, dat, []).
det(die, def, f, akk, []).
det(das, def, n, nom, []).
det(des, def, n, gen, []).
det(dem, def, n, dat, []).
det(das, def, n, akk, []).
det(die, def, pl, nom, []).
det(der, def, pl, gen, []).
det(den, def, pl, dat, []).
det(die, def, pl, akk, []).
```

［不定冠詞］

```
det(ein, indef, m, nom, []).
det(eines, indef, m, gen, []).
det(einem, indef, m, dat, []).
det(einen, indef, m, akk, []).
det(eine, indef, f, nom, []).
det(einer, indef, f, gen, []).
det(einer, indef, f, dat, []).
det(eine, indef, f, akk, []).
det(ein, indef, n, nom, []).
det(eines, indef, n, gen, []).
```

det(einem, indef, n, dat, []).
det(ein, indef, n, akk, []).

次に定冠詞類および不定冠詞類の辞書記載項（述語 det/3）と形態素規則（述語 det/5）を掲げる。この述語 det/5 のそれぞれの引数の意味は，上の定冠詞・不定冠詞のそれと同じである。形態素規則については詳述は避けるが，先の動詞の人称変化の場合と同様の語尾の操作で，上から順に男性１格から複数４格までの格変化を記述している。それぞれの節に附したコメントを参考にプログラムの動きを確認していただきたい。

[定冠詞類・不定冠詞類の辞書記載項]

det(mein, indef, 私の).
det(dein, indef, 君の).
det(sein, indef, 彼の).
det(ihr, indef, 彼女の).
det(sein, indef, それの).
det(unser, indef, 私たちの).
det(euer, indef, 君たちの).
det(ihr, indef, 彼らの).
det('Ihr', indef, あなたの).
det(kein, indef, neg).

det(dies, def, この).
det(jen, def, あの).
det(all, def, すべての).
det(welch, def, どの).
det(manch, def, かなり多くの).

```
det(jed, def, おのおのの).

[定冠詞類・不定冠詞類の形態素規則]
det(Word, indef, m, nom, Jpn) :-
        det(Word, indef, Jpn).                    %不定冠詞類男性 1 格
det(Word, def, m, nom, Jpn) :-
        name(Word, WCode),
        append(LCode, [101, 114], WCode),
        LCode ¥== [],
        name(Lex, LCode),
        det(Lex, def, Jpn).                       %定冠詞類男性 1 格
det(Word, Typ, m, gen, Jpn) :-
        name(Word, WCode),
        append(LCode, [101, 115], WCode),
        LCode ¥== [],
        name(Lex, LCode),
        det(Lex, Typ, Jpn).                       %定・不定冠詞類男性 2 格
det(Word, Typ, m, dat, Jpn) :-
        name(Word, WCode),
        append(LCode, [101, 109], WCode),
        name(Lex, LCode),
        det(Lex, Typ, Jpn).                       %定・不定冠詞類男性 3 格
det(Word, Typ, m, akk, Jpn) :-
        name(Word, WCode),
        append(LCode, [101, 110], WCode),
        name(Lex, LCode),
        det(Lex, Typ, Jpn).                       %定・不定冠詞類男性 4 格
```

```
det(Word, Typ, f, nom, Jpn) :-
    name(Word, WCode),
    append(LCode, [101], WCode),
    name(Lex, LCode),
    det(Lex, Typ, Jpn).                    %定・不定冠詞類女性 1 格
det(Word, Typ, f, gen, Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 114], WCode),
    LCode ¥== [],
    name(Lex, LCode),
    det(Lex, Typ, Jpn).                    %定・不定冠詞類女性 2 格
det(Word, Typ, f, dat, Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 114], WCode),
    LCode ¥== [],
    name(Lex, LCode),
    det(Lex, Typ, Jpn).                    %定・不定冠詞類女性 3 格
det(Word, Typ, f, akk, Jpn) :-
    name(Word, WCode),
    append(LCode, [101], WCode),
    name(Lex, LCode),
    det(Lex, Typ, Jpn).                    %定・不定冠詞類女性 4 格
det(Word, indef, n, nom, Jpn) :-
    det(Word, indef, Jpn).                 %不定冠詞類中性 1 格
det(Word, def, n, nom, Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 115], WCode);
```

```
        LCode ¥== [],
        name(Lex, LCode),
        det(Lex, def, Jpn).                    %定冠詞類中性 1 格
det(Word, Typ, n, gen, Jpn) :-
        name(Word, WCode),
        append(LCode, [101, 115], WCode),
        LCode ¥== [],
        name(Lex, LCode),
        det(Lex, Typ, Jpn).                    %定・不定冠詞類中性 2 格
det(Word, Typ, n, dat, Jpn) :-
        name(Word, WCode),
        append(LCode, [101, 109], WCode),
        name(Lex, LCode),
        det(Lex, Typ, Jpn).                    %定・不定冠詞類中性 3 格
det(Word, indef, n, akk, Jpn) :-
        det(Word, indef, Jpn).                 %不定冠詞類中性 4 格
det(Word, def, n, akk, Jpn) :-
        name(Word, WCode),
        append(LCode, [101, 115], WCode),
        LCode ¥== [],
        name(Lex, LCode),
        det(Lex, def, Jpn).                    %定冠詞類中性 4 格
det(Word, Typ, pl, nom, Jpn) :-
        name(Word, WCode),
        append(LCode, [101], WCode),
        name(Lex, LCode),
        det(Lex, Typ, Jpn).                    %定・不定冠詞類複数 1 格
```

```
det(Word, Typ, pl, gen, Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 114], WCode),
    LCode ¥== [],
    name(Lex, LCode),
    det(Lex, Typ, Jpn).                    %定・不定冠詞類複数2格
det(Word, Typ, pl, dat, Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 110], WCode),
    name(Lex, LCode),
    det(Lex, Typ, Jpn).                    %定・不定冠詞類複数3格
det(Word, Typ, pl, akk, Jpn) :-
    name(Word, WCode),
    append(LCode, [101], WCode),
    name(Lex, LCode),
    det(Lex, Typ, Jpn).                    %定・不定冠詞類複数4格
```

名詞の形態素規則の場合は，複数形の処理および男性・中性単数2格や複数3格，男性弱変化名詞の格語尾の処理が問題となる。複数形については，英語の場合のように一つの語尾パターンではなく，複数のパターンがあり，また語幹が変母音するものも多く，形態素規則として定式化することは難しい。そこで本システムでは，以下の名詞の語彙記載項のプログラム例に見られるように，2格の変化語尾をも含めて，複数形を語彙項目の中に直接記述する方法をとっている。述語 n/5 の第3引数がそれである。この引数は二つの項からなるリストの形になっており，第1項が単数2格形，第2項が複数形を表わす。単数2格形については，語尾のパターンに -s, -es, -n, -en があり，複数形と同じ理由で形態素規則の導入が難しいため，同様の処理にし

た。これらの項は，以下で述べる格変化の形態素規則に引き渡され，ユニフィケーションが実行されることになる。なおその他の引数についてふれておくと，第1引数は語形，第2引数は性，第4引数は後の構文解析において必要となる意味素性を表わし，第5引数は日本語訳である。

[名詞の語彙記載項の例]

n('Arzt', m, [['Arztes'], 'Aerzte'], human, 医者).
n('Ball', m, [['Balls'], 'Baelle'], konkr, ボール).
n('Deutsch', n, [[[], []], []], abstr, ドイツ語).
n('Englisch', n, [[[], []], []], abstr, 英語).
n('Frau', f, ['Frauen'], human, 女).
n('Gebaeude', n, [['Gebaeudes'], 'Gebaeude'], konkr, 建物).
n('Japanisch', n, [[[], []], []], abstr, 日本語).
n('Kasten', m, [['Kastens'], 'Kasten'], konkr, 箱).
n('Kind', n, [['Kindes'], 'Kinder'], human, 子供).
n('Lehrer', m, [['Lehrers'], 'Lehrer'], human, 先生).
n('Mann', m, [['Manns', 'Mannes'], 'Maenner'], human, 男).
n('Strasse', f, ['Strassen'], konkr, 通り).
n('Student', m, [['Studenten'], 'Studenten'], human, 学生).
n('Vater', m, [['Vaters'], 'Vaeter'], human, 父).
n('Wetter', n, [['Wetters'], 'Wetter'], abstr, 天気).

次に名詞の格変化の形態素規則を掲げる。述語 noun/5 が，節の本体の述語 n/5 において上で述べた名詞の各語彙項目とユニファイすることにより，語形の性・数・格を決定することになる。

[名詞格変化の形態素規則]

```
noun(Word, Sex, nom, Sem, Jpn) :-
    n(Word, Sex, [[G], Pl], Sem, Jpn).
noun(Word, Sex, nom, Sem, Jpn) :-
    n(Word, Sex, [[G1, G2], Pl], Sem, Jpn).        %単数1格（男・中）
noun(Word, f, nom, Sem, Jpn) :-
    n(Word, f, [Pl], Sem, Jpn).                    %単数1格（女）
noun(Word, Sex, gen, Sem, Jpn) :-
    n(D, Sex, [[Word, G2], Pl], Sem, Jpn).
noun(Word, Sex, gen, Sem, Jpn) :-
    n(D, Sex, [[G1, Word], Pl], Sem, Jpn).
noun(Word, Sex, gen, Sem, Jpn) :-
    n(D, Sex, [[Word], Pl], Sem, Jpn).             %単数2格（男・中）
noun(Word, f, gen, Sem, Jpn) :-
    n(Word, f, [Pl], Sem, Jpn).                    %単数2格（女）
noun(Word, Sex, dat, Sem, Jpn) :-
    n(Word, Sex, [[G1, G2], Pl], Sem, Jpn).
noun(Word, Sex, dat, Sem, Jpn) :-
    n(Word, Sex, [[G], Pl], Sem, Jpn),
    name(G, GCode),
    ¥+append(SCode, [110], GCode).                 %単数3格（男・中）
noun(Word, m, dat, Sem, Jpn) :-
    n(WD, m, [[Word], Pl], Sem, Jpn),
    name(Word, WCode),
    append(SCode, [110], WCode).                   %単数3格（男性弱変化）
noun(Word, f, dat, Sem, Jpn) :-
    n(Word, f, [Pl], Sem, Jpn).                    %単数3格（女）
```

```
noun(Word, Sex, akk, Sem, Jpn) :-
    n(Word, Sex, [[G], Pl], Sem, Jpn),
    name(G, GCode),
    ¥+append(SCode, [110], GCode).
noun(Word, Sex, akk, Sem, Jpn) :-
    n(Word, Sex, [[G1, G2], Pl], Sem, Jpn).        %単数 4 格（男・中）
noun(Word, m, akk, Sem, Jpn) :-
    n(WD, m, [[Word], Pl], Sem, Jpn),
    name(Word, WCode),
    append(SCode, [110], WCode).                   %単数 4 格（男性弱変化）
noun(Word, f, akk, Sem, Jpn) :-
    n(Word, f, [G], Sem, Jpn).                     %単数 4 格（女）

noun(Word, pl, nom, Sem, Jpn) :-
    n(WD, Sex, [G, Word], Sem, Jpn).               %複数 1 格（男・中）
noun(Word, pl, gen, Sem, Jpn) :-
    n(WD, Sex, [G, Word], Sem, Jpn).               %複数 2 格（男・中）
noun(Word, pl, akk, Sem, Jpn) :-
    n(WD, Sex, [G, Word], Sem, Jpn).               %複数 4 格（男・中）
noun(Word, pl, nom, Sem, Jpn) :-
    n(WD, f, [Word], Sem, Jpn).                    %複数 1 格（女）
noun(Word, pl, gen, Sem, Jpn) :-
    n(WD, f, [Word], Sem, Jpn).                    %複数 2 格（女）
noun(Word, pl, akk, Sem, Jpn) :-
    n(WD, f, [Word], Sem, Jpn).                    %複数 4 格（女）
noun(Word, pl, dat, Sem, Jpn) :-
    (name(Word, D),
```

```
        append(E, [110], D),
        name(F, E),
        n(G, Sex, [H, F], Sem, Jpn)
    ; n(G, Sex, [H, Word], Sem, Jpn)
    ).                                    %複数3格（男・中）
noun(Word, pl, dat, Sem, Jpn) :-
    (name(Word, D),
        append(E, [110], D),
        name(F, E),
        n(G, f, [F], Sem, Jpn)
    ; n(G, f, [Word], Sem, Jpn)
    ).                                    %複数3格（女）
```

代名詞に関しては，いわゆる指示代名詞（dies-, jed- usw.），不定代名詞（all-, manch- usw.），関係代名詞など，冠詞とほぼ同様の格変化語尾をとるものについては，冠詞と異なる語尾パターンを示す部分についてのみ付加的な形態素規則を導入すればよい。これらが単独で名詞句として機能することは，後の構文解析の段階で規則化される。語尾の変化ではなく，語彙的な変化形をもつ人称代名詞や再帰代名詞，疑問代名詞等については，それぞれ独立の語彙項目として記載しておく必要がある。一例として，以下に人称代名詞の語彙記載例（述語名は pron/6）を掲げる。それぞれの引数は，第1引数から，語形，数，人称，格，代名詞のクラス（per は人称代名詞を示す），日本語訳を表わす。最後にある敬称2人称の 'Sie' は数，人称が3，pl（3人称複数）となっているが，これは後でふれることになる Kongruenz の問題と関連している。つまり例えば主格の 'Sie' が定形動詞と結びつくとき，定形動詞の人称変化語尾が，3人称複数と同じであり，こうしておくことによって動詞の形態素規則を余分に記述することを避けることができる。

[人称代名詞の語彙記載項]

pron(ich, sg, 1, nom, per, 私).

pron(mir, sg, 1, dat, per, 私).

pron(mich, sg, 1, akk, per, 私).

pron(du, sg, 2, nom, per, 君).

pron(dir, sg, 2, dat, per, 君).

pron(dich, sg, 2, akk, per, 君).

pron(er, sg, 3, nom, per, 彼).

pron(ihm, sg, 3, dat, per, 彼).

pron(ihn, sg, 3, akk, per, 彼).

pron(sie, sg, 3, nom, per, 彼女).

pron(ihr, sg, 3, dat, per, 彼女).

pron(sie, sg, 3, akk, per, 彼女).

pron(es, sg, 3, nom, per, それ).

pron(ihm, sg, 3, dat, per, それ).

pron(es, sg, 3, akk, per, それ).

pron(wir, pl, 1, nom, per, 私たち).

pron(uns, pl, 1, dat, per, 私たち).

pron(uns, pl, 1, akk, per, 私たち).

pron(ihr, pl, 2, nom, per, 君たち).

pron(euch, pl, 2, dat, per, 君たち).

pron(euch, pl, 2, akk, per, 君たち).

pron(sie, pl, 3, nom, per, 彼ら).

pron(ihnen, pl, 3, dat, per, 彼ら).

pron(sie, pl, 3, akk, per, 彼ら).

pron('Sie', pl, 3, nom, per, あなた).

pron('Ihnen', pl, 3, dat, per, あなた).

pron('Sie', pl, 3, akk, per, あなた).

格変化詞の形態素規則の最後に，最も複雑な変化のパターンを示す形容詞を取り上げよう。形容詞の場合は，比較変化および格変化の二つの規則が必要である。いずれも先の動詞および冠詞の場合と同様，処理は語尾の操作である。同一規則の繰り返しで，特に格変化の形態素規則については相当煩雑なものになるが，ここではそのうち混合変化の全規則を掲げておくことにする。形容詞の語彙項目は，述語 adj/3 によって記述される。例によって第1引数は語形，最後の引数は日本語訳である。第2引数は，統語素性の指定で，adv は副詞的用法が可能，non は不可能なものを示す。比較変化の形態素規則で得られる比較級，最高級の語形は，語彙項目と同じく述語 adj/3 で表わされる。格変化の形態素規則では，変化語形は述語 adj_attr/7 で表わされるが，ここでは差分リストを用いた表現にしている。すなわち第1引数と第2引数がそれで，第3引数以下，変化系列（強変化—indef0，弱変化—def，混合変化—indef），性，格，パースツリーを作るための部分木，日本語訳を示す。この日本語訳は，それぞれの節の本体の最後に導入される述語 adj/3 の第3引数と同一の変数であり，これにより語彙項目に記述されている日本語訳の情報が引き渡されることになる。

［形容詞の語彙記載項］

adj(alt, non, 古い).

adj(interessant, non, おもしろい).

adj(neu, non, 新しい).

adj(schoen, adv, 美しい).

adj(schwierig, adv, 難しい).

```
[形容詞の比較変化規則]
  adj(Word, Typ, [より, Jpn]) :-
        name(Word, WCode),
        append(LCode, [101, 114], WCode),
        name(Lex, LCode),
        adj(Lex, Typ, Jpn).                          %比較級
  adj(Word, Typ, [もっとも, Jpn]) :-
        name(Word, WCode),
        append(LCode, [115, 116], WCode),
        name(Lex, LCode),
        adj(Lex, Typ, Jpn).                          %最高級

[形容詞（混合変化）の格変化規則]
  adj_attr([Word|X], X, indef, m, nom, [adj, Word], Jpn) :-
        name(Word, WCode),
        append(LCode, [101, 114], WCode),
        LCode ¥== [],
        name(Lex, LCode),
        adj(Lex, Typ, Jpn).                          %男性 1 格
  adj_attr([Word|X], X, indef, m, gen, [adj, Word], Jpn) :-
        name(Word, WCode),
        append(LCode, [101, 110], WCode),
        name(Lex, LCode),
        adj(Lex, Typ, Jpn).                          %男性 2 格
  adj_attr([Word|X], X, indef, m, dat, [adj, Word], Jpn) :-
        name(Word, WCode),
        append(LCode, [101, 110], WCode),
```

```
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %男性 3 格
adj_attr([Word|X], X, indef, m, akk, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 110], WCode),
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %男性 4 格
adj_attr([Word|X], X, indef, f, nom, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101], WCode),
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %女性 1 格
adj_attr([Word|X], X, indef, f, gen, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 110], WCode),
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %女性 2 格
adj_attr([Word|X], X, indef, f, dat, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 110], WCode),
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %女性 3 格
adj_attr([Word|X], X, indef, f, akk, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101], WCode),
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %女性 4 格
```

```
adj_attr([Word|X], X, indef, n, nom, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 115], WCode),
    LCode ¥== [],
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %中性 1 格
adj_attr([Word|X], X, indef, n, gen, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 110], WCode),
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %中性 2 格
adj_attr([Word|X], X, indef, n, dat, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 110], WCode),
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %中性 3 格
adj_attr([Word|X], X, indef, n, akk, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 115], WCode),
    LCode ¥== [],
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %中性 4 格
adj_attr([Word|X], X, indef, pl, nom, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 110], WCode),
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %複数 1 格
```

```
adj_attr([Word|X], X, indef, pl, gen, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 110], WCode),
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %複数 2 格
adj_attr([Word|X], X, indef, pl, dat, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 110], WCode),
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %複数 3 格
adj_attr([Word|X], X, indef, pl, akk, [adj, Word], Jpn) :-
    name(Word, WCode),
    append(LCode, [101, 110], WCode),
    name(Lex, LCode),
    adj(Lex, Typ, Jpn).                          %複数 4 格
```

3. Kongruenz の問題

以上ドイツ語の形態素規則について多少詳しく検討してきたが，これらの規則によって解析された変化語形は，その出現に一定の統語的な制限がある。いわゆる Kongruenz（一致）と呼ばれる文法現象である。これは本来，むしろ次回に取り上げることになる構文解析の問題であると言えるが，以上の形態素規則と密接に関わっている問題でもあり，形態素規則の記述のまとめとして，ここで取り上げることにする。この現象は，ドイツ語においては，主語と定動詞（ich singe, du singst ...）および名詞句における冠詞，形容詞，名詞間（die alte Frau, der alten Frau ...）に見られる。前者の場合は人称と数の一致であり，後者においては性・数・格の一致である。これ

らの現象が適切に規則化されないと不適格文を排除できないわけで，解析システムの根幹に関わる問題とも言える。他の手続き型言語では処理が相当面倒なものとなるこの現象を，Prolog では，その動作の基本原理であるユニフィケーションの手法で，いとも簡単に解決する。次のプログラムは，次回で取り上げる構文解析プログラムの抜粋（正置平叙文と名詞句の一部）であるが，これによってその動きを確認しておこう。

［正置平叙文の句構造規則］

```
s_dec(A, [], [s, T1|T2], [J1, は, J2]) :-
        append(A1, [V|A2], A),
        A1 ¥== [],
        np(A1, [], Num, Psn, Sex, nom, Sem, T1, J1),
        v([V], [], Typ, Num, Psn, _, _, _, _, _),
        append(A2, [V], B),
        s_prd(B, [], Typ, Num, Psn, T2, J2).
```

［名詞句の句構造規則（一部）］

```
np0(A, B, Num, 3, Sex, Case, Sem, [Case, T1, T2], [J1, J2]) :-
        det(A, X, Typ, Num, Sex, Case, T1, J1),
        nominal(X, B, Typ, Num, Sex, Case, Sem, T2, J2).

nominal(A, B, Typ, Num, Sex, Case, Sem, [n, T1, T2], [J1, な, J2]) :-
        adj_attr(A, X, Typ, Sex, Case, T1, J1),
        noun(X, B, Num, Sex, Case, Sem, T2, J2).
```

最初の正置平叙文のプログラムは，概略，「正置平叙文は，『１格名詞＋定動詞＋後続要素』という連鎖であり，この後続要素の後に定動詞を付加した

ものが動詞句として以降の解析プロセスに引き渡される」というものである。プログラムの細部の動きは次回に譲るとして，ここで注目すべき点は，4行目と5行目の述語 np/9 および v/11 である。ここでは両述語で同一の変数名が使われている部分がある。n/9 の第3引数と v/11 の第4引数の Num および n/9 の第4引数と v/11 の第5引数の Psn がそれである。この処置によって，変数にマッチする定数が両述語で同じものでなければならない，という制限が設けられ，この制限に合わないものは排除されることになる。ここでは変数 Num は数を，変数 Psn は人称を表わし，したがって数と人称の一致が記述されているわけである。例えば ich singe と *ich singt の場合を取り上げると，

```
                ⋮
np([ich], [ ], sg, 1, Sex, nom, Sem, T1, J1),
v([singe], [ ], i, sg, 1, _, _, _, _, _, _),
                ⋮
                ⋮
np([ich], [ ], sg, 1, Sex, nom, Sem, T1, T2),
v([singt], [ ], i, sg, 3, _, _, _, _, _, _),
                ⋮
```

のようになり，前者は Num, Psn ともマッチした定数の値が等しくユニフィケーションが成功し，後者は Psn の値が異なるため，失敗することになる。

名詞句の句構造規則は，述語 np0/9 および nominal/9 の二段構えになっているが，まず np0/9 で，名詞句は det/8（冠詞）と非終端記号 nominal/8 に分析され，nominal/8 はさらに，adj_attr/7（付加語形容詞）と noun/8 に分析される。詳細は省くが，それぞれの節で同名の変数 Num, Sex, Case

が各述語に受け渡されている事実を確認していただきたい。これにより，先の主語と定動詞の場合と同様，名詞句内部の各要素の数（Num），性（Sex），格（Case）の一致が記述されているわけである。

〔注〕

1） 本稿は，昭和62年度科学研究費補助金奨励研究（A）課題番号62710277「ドイツ語の機械解析に向けての予備的研究」の研究成果の一部である。

2） 坂本義行「実用化された機械翻訳」（月刊『言語』1988年1月号，p. 60-67）参照。

3） 動作環境は，ハードウェア本体 PC9801-VX21（日本電気）にRAMボード SB-2000（メルコ）を装着したもの。ソフトウェアは Arity Prolog Compiler Ver. 5.0（ライフボート）。RAMボードはRAMディスクとして使用。これはソフトウェアが仮想記憶によるデータベース管理をおこなっているため，ある程度以上大きなプログラムになると，フロッピーディスク・ベースでは実用的なスピードを維持できないことに対する処置である。

4） 例えば，高木朗・伊東幸宏『自然言語の処理』（情報処理実用シリーズ10）丸善，1987年，参照。

5） 選択制限（selectional restriction）は，生成変形文法の用語。Vgl. Chomsky, N.: Aspects of the Theory of Syntax. Cambridge, Mass. 1965. 分布（distribution）は，元来，アメリカ構造主義言語学の概念であるが，東ドイツの Helbig たちの手になる『結合価と分布の辞典』シリーズで広く知られている。Vgl. Helbig, G. / W. Schenkel: Wörterbuch zur Valenz und Distribution deutscher Verben. Leipzig 1969. Sommerfeldt, K.-E. / H. Schreiber: Wörterbuch zur Distribution deutscher Adjektive. Leipzig 1974. Sommerfeldt, K.-E. / H. Schreiber: Wörterbuch zur Valenz und Distribution deutscher Substantive. Leipzig 1977.

6） Prolog 言語の概要については，各ソフトウェア添付のマニュアルの他，現在数多くの入門書・解説書が出版されており，それらを参照のこと。

7） Prolog では，英小文字や数字で始まるアトムは定数となる。また変数アトムは英大文字で始めなければならない。

8） パターン・マッチングのことを，Prolog ではとくにユニフィケーションまたは単一化と呼ぶ。

9） この述語 append/3 は組み込み述語ではない。Prolog での基本操作に欠かせない述語で，どの参考書でも取り上げられているが，参考までにそのプログラムを以下に示す。

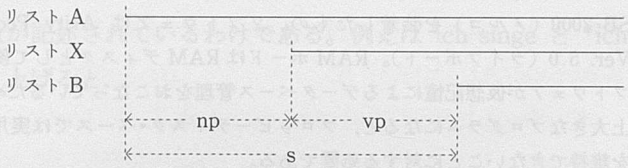
append([ ], X, X).

append([A|X], Y, [A|Z]) :-append(X, Y, Z).

10) 二つのリストの差で一つのリストを表わす手法。例えば，([a, b|X], X) というリストの対でリスト [a, b] が表わされる。この方法だと，あるリストを二つ以上に分解したい場合，リストの差をポインタとして使えるので，文の構成要素の解析の場合のように分解したい位置が不定の場合，上述の述語 append/3 を使う場合に比べて解析の効率が格段によいことが知られている。例えば文 (s) を名詞句 (np) と動詞句 (vp) に分解する場合，差分リストによる表現では次のようになる。

s(A, B) :-np(A, X), vp(X, B).

分かりやすく図式化すると，



のようになり，各リストの差で構成要素が表現されている様子が理解できよう。Prolog における標準的なトップダウン・パーザーである DCG（限定節文法）はこの手法を用いている。