岐阜経済大学論集 28 巻 4 号（1995 年）

〔研究ノート〕

ドイツ語の機械解析に向けて (2)

山 田 善 久

4. ドイツ語の構文解析

自然言語の構文解析を，局所的な例文にとどまらず，その言語で可能な文を網羅するような一般性をもったものにするためには，基礎となる統語理論について，ある程度原理的な考察が必要となる。その場合純粋に理論的立場からではなく，コンピュータ（とくにソフトウェア）の動作原理を踏まえて，もっとも効率的な手法を取捨選択しなければならない。いわば機械との妥協であるが，一方で，ここで実現されたものが理論そのものにフィードバックされ得る契機もあるように思われる。

具体的には，まず，最近のドイツ語生成統語論の枠内で議論されているような[1]，ドイツ文の句構造は階層的タイプなのか，それともフラットなタイプなのか，という問題がある（4.1で扱う）。次にPrologという処理系を前提にして，ドイツ文の構文解析をおこなう場合，この処理系の動作原理と，これとは必ずしもうまくフィットしないドイツ語特有のいくつかの統語論的特性を，どのように機械とチューニングさせていくかという問題がある。この関連での主なポイントは，語順に関するもので，ドイツ語のいわゆる枠構造（助動詞構文や分離動詞）に見られる述語の不連続要素の処理（4.2で扱う），および比較的自由な文成分の配列順序の処理（4.3で扱う）の二点である。

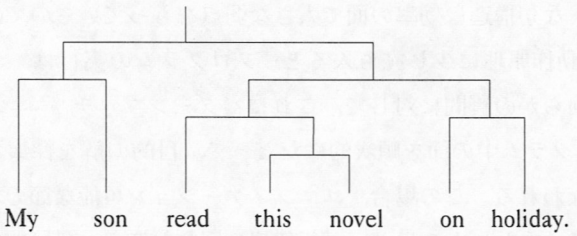
以下，4.1，4.2は，ドイツ語の構文解析の一般的方策についての検討である。4.3で，これらを踏まえながら，具体的にPrologによる構文解析規則を導入する。

4.1. 基底句構造の構想

19世紀以来伝統の形態論中心の文法記述から脱却して，シンタックス（統語論）の重要性が広く意識されるようになったのは，20世紀の構造主義，とくにプラーグ学派のいわゆる機能的文構成（Funktionale Satzperspektive）の研究の影響によるところが大きいと思われる。ドイツ文法論の枠内で，その先鞭をつけたのは，E. Drach（1937）であった。このDrach以来のドイツ文論は，Erben（1968）やSchulz-Griesbach（1960）のようなどちらかといえば教育的・機能主義的立場の著作にせよ，比較的最近のHelbig / Buscha（1972）やEngel（1988）らValenz理論ないしは依存関係理論的立場の著作にせよ，ドイツ文の基本構造として想定しているのは，述語（動詞）を中心とする文成分（Satzglied）の概念である。この立場では，用語に多少の差異はあるものの，主語（Subjekt），目的語（Objekt），補足語（Ergänzung），状況語（Angabe）などの文成分が中核にある述語（Prädikat）の支配を受けており，個々の文成分自体は互いに対等の関係にあるものと考える。つまりフ

(1a) Mein Sohn las diesen Roman im Urlaub.

(1b)



ラットな句構造を想定している――(1a)。

一方，英文法系では，直接構成要素 (IC) 分析や Chomsky の初期生成文法以来の句構造規則のように，主部・述部の二分法から出発し，述部をさらに動詞句，名詞句のようなより小さな句構造に分析していき，終端の単語のレベルに至るといった，階層構造をもった文分析法が主流をなしている。(1b)に見られるように，この分析法は，ドイツ系文論のフラットな句構造の構想とは対照的である。

この相違は，それぞれの言語の現実に即した観察から必然的に出てきた結果とみなすこともできる。ドイツ語はフラットな句構造をもち，英語は階層的な句構造をもつというわけで，個別言語の多様性を示す一つの事例と考えられなくもない。言語の普遍性を標榜し，表層では多様な姿を示す諸言語も，深層では同様の基底構造をもつという仮定をいわば拠り所にしていた生成文法も，最近の GB 理論では，パラメータというかたちで，個別言語の多様性を説明する装置を，理論の中に組み入れている。

ともあれ機械に乗せるという観点からは，ドイツ語がフラットな構造をもつ言語という理論的前提をそのまま踏襲してよいかどうかは，あらためて検討してみる必要がある。この機械によるドイツ文解析の試みも，当初はフラ

ットな構造を想定して構文規則をコーディングしていったのであるが，途中で大きな壁にぶつかった。これはPrologという処理系の動作原理にかかわるテクニカルな問題であるが，とくに本試みのように，解析がトップダウンで，つまり大きな構成要素から小さな構成要素に向かっておこなわれる場合，フラットな句構造は効率の面で大きな弱点をもっているのである。

Prologの動作原理に少し立ち入ると，プログラムの実行は，ユーザから与えられた何らかの質問に対して，これにパターンマッチ（ユニフィケーション）するプログラム中の節を順次的にたどって，目的の解を探索するというふうにおこなわれる。この場合，ユニフィケーション可能な節のオールタネイティブが多くなればなるほど，試行錯誤の回数が増え，最終的な解にたどりつくまでに多くの時間を要する。この時間は，例えば通常のデータベースのようなプログラムであれば，ほとんど無視できるものであるが，自然言語の解析のような場合には，かなり深刻な問題となる。さまざまな文型，さまざまな句の内部構造を網羅的に処理するための複雑な構文規則が必要であり，これに膨大な辞書部門が加わる。プログラムの探索経路も，通常のプログラムのように単純ではなく，大量の構文規則群の中を，プログラマ自身にも見通しがつかないほど複雑に駆けめぐる。トップダウンの解析の場合だと，最初にマッチングを試みるのは文のレベルであり，文から別の文へ……そこから下の階層の句へ……句から別の句へ……さらにはその下の階層の句へ……と，現場に立ち会った経験がないと想像がつかないと思われるが，この間のコンピュータの演算の回数はほとんど天文学的と言っても過言ではない。Prologマシンや LISPマシンといった論理型言語に特化したハードの開発や，BUPのような新しい統語解析アルゴリズムの提案[2]も，こうした論理型言語による自然言語処理の想像を絶する複雑さの改善をめざす方向から出たものといえよう。

こうした情報工学の分野での展開はともかくとして，当面する問題は，現在の標準Prologという環境で，いかに処理の効率化を図るかということで

ある。そしてこの検討は，ハードウェア/ソフトウェアの今後の発展によって解消されていくようなものではなく，解析の効率化を考える上で常に押さえておくべきポイントとなるように思われる。その原則は，言ってしまえばごく単純で，「ユニフィケーション可能な節のオールタネイティブを極力減らす」，ということである。このことは，プログラムが最初にユニフィケーションを試みるトップエンドにある節に対してとくに当てはまる。トップダウン解析の場合だと，始発の文のレベルに相当する節がこれに該当する。

ここで先ほどのフラットか階層的かという句構造の議論に戻ると，フラットな句構造を想定した場合，動詞に結合する要素（文成分）の数や種類に応じて複数個の，場合によっては数十個もの文型を設定するといった作業が必要となる。例えばこのような文型論の典型として，Duden-Grammatik (1984) の記述を見てみよう。表1に見られるように，ここでは37個の文型が設定されている。これらを細かく見ると，そこには ... Objekt, ... Ergänzung といった細分化された機能的範疇が含まれており，その細分化に応じて文型の数が多くなっているのであるが，それらをすべて無視して，単に Prädikat に結びつく要素の数だけで文型を再分類すると，要素が Subjekt だけの1の場合から，最大4まで4通りに集約することができる。しかし機械にかけるという観点から見れば，これでもまだ多いのである。ここには動詞の支配を受けない随意的な副詞規定語，つまり状況語（Angabe）が含まれていないし，疑問文などの文タイプの変容を含めると，その数倍の文型が必要となるだろう。これは望ましいことではない。とくに一つの文に含め得る状況語の数は，実際には言語運用上の制約はあるにしても，理論上は無限である。

階層的な句構造規則を用いると，文型の数の問題，つまりオールタネイティブの問題が解消される。この場合，規則のトップエンドは，たった一つのS（文）で済んでしまうからである。このSをさらに展開していく中で，VP→NP VP といった再帰的な規則をも交えながら，要素の個数の異なる

表1 Duden-Grammatik (1984) S. 635

1. Subjekt＋Prädikat	*Die Rosen blühen.*
2. Subjekt＋Prädikat＋Akkusativobjekt	*Der Gärtner bindet die Blumen.*
3. Subjekt＋Prädikat＋Dativobjekt	*Der Sohn dankt dem Vater.*
4. Subjekt＋Prädikat＋Genitivobjekt	*Ich harre seiner.*
5. Subjekt＋Prädikat＋Präpositionalobjekt	*Inge achtet auf ihre Schwester.*
6. Subjekt＋Prädikat＋Gleichsetzungsnominativ	*Karl ist mein Freund.*
7. Subjekt＋Prädikat＋Raumergänzung	*Das Buch liegt auf dem Tisch.*
8. Subjekt＋Prädikat＋Zeitergänzung	*Die Beratung dauerte zwei Stunden.*
9. Subjekt＋Prädikat＋Artergänzung	*Die Rose ist schön.*
10. Subjekt＋Prädikat＋Begründungsergänzung	*Das Verbrechen geschah aus Eifersucht.*
11. Subjekt＋Prädikat＋Dativobjekt＋Akkusativobjekt	*Werner schenkt seiner Mutter Blumen.*
12. Subjekt＋Prädikat＋Akkusativobjekt＋Genitivobjekt	*Der Richter beschuldigte ihn des Diebstahls.*
13. Subjekt＋Prädikat＋Akkusativobjekt＋Präpositionalobjekt	*Er verriet ihn an seine Feinde.*
14. Subjekt＋Prädikat＋Akkusativobjekt＋Raumergänzung	*Ich hänge das Bild an die Wand.*
15. Subjekt＋Prädikat＋Akkusativobjekt＋Zeitergänzung	*Er zog das Gespräch in die Länge.*
16. Subjekt＋Prädikat＋Akkusativobjekt＋Artergänzung	*Die Mutter macht die Suppe warm.*
17. Subjekt＋Prädikat＋Artergänzung＋Präpositionalobjekt	*Er handelt niederträchtig an ihm.*
18. Subjekt＋Prädikat＋Artergänzung＋Raumergänzung	*Es geht lustig zu auf der Festwiese.*
19. Subjekt＋Prädikat＋Akkusativobjekt＋Gleichsetzungsakkusativ	*Klaus nennt mich einen Lügner.*
20. Subjekt＋Prädikat＋Akkusativobjekt＋Akkusativobjekt	*Herr Meier lehrt uns die französische Sprache.*
21. Subjekt＋Prädikat＋Dativobjekt＋Präpositionalobjekt	*Ich rate dir zum Nachgeben.*
22. Subjekt＋Prädikat＋Dativobjekt＋Artergänzung	*Es geht mir schlecht.*
23. Subjekt＋Prädikat＋Präpositionalobjekt＋Präpositionalobjekt	*Er sprach zu den Kindern über seine Reise.*
24. Subjekt＋Prädikat＋Artergänzung＋Dativobjekt (2. Grades)	*Ich bin diesem Mann fremd.*
25. Subjekt＋Prädikat＋Artergänzung＋Genitivobjekt (2. Grades)	*Er ist des Diebstahls schuldig.*
26. Subjekt＋Prädikat＋Artergänzung＋Präpositionalobjekt (2. Grades)	*Ich bin auf deinen Bericht gespannt.*
27. Subjekt＋Prädikat＋Artergänzung＋Dativobjekt (2. Gd.)＋Präp.-Obj. (2. Gd.)	*Er ist mir an Ausdauer überlegen.*
28. Subjekt＋Prädikat＋Artergänzung＋Raumergänzung (2. Grades)	*Er ist in München ansässig.*
29. Subjekt＋Prädikat＋Akkusativobjekt (2. Gd.)＋Artergänzung	*Der Spalt ist einen Fuß breit.*
30. Subjekt＋Prädikat＋Akk.-Obj. (1. Gd.)＋Akk.-Obj. (2. Gd.)＋Artergänzung	*Er wirft den Ball 70 m weit.*
31. Subjekt＋Prädikat＋Akkusativobjekt (2. Gd.)＋Raumergänzung	*Er geht die Treppe hinunter.*
32. Subjekt＋Prädikat＋Akk.-Obj. (1. Gd.)＋Akk.-Obj. (2. Gd.)＋Raumergänzung	*Sie warfen ihn die Treppe hinunter.*
33. Subjekt＋Prädikat＋Pertinenzdativ	*Dem Kind blutet die Hand.*
34. Subjekt＋Prädikat＋Akkusativobjekt＋Pertinenzdativ	*Er streichelt ihr die Wangen.*
35. Subjekt＋Prädikat＋Akkusativobjekt＋Arterg.＋Pertinenzdativ	*Der Arzt richtet ihr die Nase gerade.*
36. Subjekt＋Prädikat＋Raumergänzung＋Pertinenzdativ	*Ich klopfe ihm auf die Schulter.*
37. Subjekt＋Prädikat＋Akkusativobjekt＋Raumergänzung＋Pertinenzdativ	*Er legt ihm die Hand auf die Schulter.*

種々の文型をすべて網羅する句構造規則を作成することが見通しとして可能となる。もちろん実際の解析システムでは，この試みのような小規模なものでも，複雑な要素をあれこれ考慮に入れなければならないので，それほど単純に事が運ぶわけではないにしても。

4.2. 不連続要素の処理

ドイツ語の不連続要素は，いわゆる枠構造，すなわち次のような助動詞や分離動詞の構文に見られる。

(2a) Hans *kann* sehr gut Tennis *spielen*.
Er *hat* gestern ein schönes Mädchen *getroffen*.
Das Mädchen *kommt* heute in Berlin *an*.

これらは，(2b)に見られるように，従属接続詞に導かれた場合，つまり副文になると，定形が後置され，述語の不連続が解消される。

(2b) ..., daß Hans sehr gut Tennis *spielen kann*.
..., daß er gestern ein schönes Mädchen *getroffen hat*.
..., daß das Mädchen heute in Berlin *ankommt*.

これらの処理をどのように考えたらよいだろうか。一つの方法は，これらの表面上の構造どおりに，主文用の，述語部分が二つの場合の構文規則と，副文用の，述語が連続した構文規則を別に導入することである。もう一つは，(2a) (2b)のどちらかを基底形とみなし，単一の構文規則を導入し，一方をその派生形として扱う方法である。この場合，派生形の述語部分を，基底形に変換する何らかの移動規則を追加する必要がある。

オールタネイティブを極力減らす，という先ほどの機械処理の原則からす

ると，前者の方法は明らかに効率が悪く，得策ではない。後者の方法も，移動規則といった新たなルールを導入するという意味では，解析システムをことさら複雑にするわけであるが，この場合は，手続き的な処理を一つ追加するだけなので，試行錯誤の繰り返しといった句構造規則の複雑さに比べれば，はるかに機械への負担が少ない。そこで本試みでは，この後者の方法を採用している。

理論的に見れば，この方法は，1960年代半ば，Chomsky理論が，まだ生成変形文法と呼ばれていた時代の枠組みに類似している。その後のこの学派の発展の中で，変形規則の理論的妥当性が疑問視され，過剰な変形の能力が次第に抑制される方向に理論が展開し，最近の一般化句構造文法（GPSG）や語彙機能文法（LFG）などのように，もはや変形規則のない生成文法が登場するに至っている。この流れからすると，本試みで採った方法は，明らかに時代逆行のように見える。GPSGやLFGは，もともと機械処理ということを念頭に置いた文法理論である。この枠組みに沿った機械へのインプリメンテーションというのが，恐らくは正しい道筋なのであろう。しかし，ここで話題にしているようなドイツ語特有の語順のふるまいを，この枠組みでどのように位置づけるかは必ずしも明確でない。この枠組みを，ある程度包括的にドイツ語分析に適用した研究は，寡聞にして知らないが，その場合もやはり，ここでの移動規則のようなマイナーなルールが，いずれにせよ必要となるのではないかという気がしている。

基底形と派生形，さらに派生形から基底形へ変換する移動規則，といった処理方法を採る場合，次に問題となるのは，どの構造を基底形と考えればよいかという点である。ここでの関連で言うと，主文のSVO構造か，それとも副文のSOV構造か，という問題である。この問題は理論文法の内部でも，とくに枠組みに大きな変化が現れた時期などに，繰り返し論議されてきたテーマである。例えば1980年代初頭にGB理論が登場するが，これに少し遅れて80年代半ば頃に，ドイツ語統語論の枠内でも，この問題が取り上

げられている[3]。機械処理の面から考えると，この問題は，どちらの方が効率的かということに帰着する。そしてこの点では，SOV 構造の有利性ははっきりしている。例をあげてみよう。

移動規則が適用されるのは，解析の過程で，定形がどれなのかが確定されたときである。その段階で，その定形の位置関係により，それが派生形と判定されれば，基底形のしかるべき位置に移動される。基底形が主文の SVO 構造の場合は，副文が派生形であり，この副文の末尾の要素（定形）が基底形である主文の定形の位置に移動される。また基底形が副文の SOV 構造の場合は，主文が派生形であり，主文中第 2 位を占める定形が，基底形である副文の定形の位置，つまり文末に移動される。要するに，移動規則を設定する際のポイントは，定形の確定と，派生形の場合はその移動，ということになるが，次にこの両者の効率を比較してみよう。

定形の確定の点では，副文の方がはるかに簡単である。例外的な逸脱文は除くと，副文では常に定形が文末に位置するからである。次にこの確定された定形を，基底形のしかるべき位置（つまり主文の定形第 2 位の位置）に移動しなければならないが，ここで難点にゆき当たる。主文が基底形とすると，移動される定形の移動地点が確定できないからである。定形第 2 位だといっても，どこが第 2 位の位置なのかは，解析が終了するまではわからない。定形の前，つまり文の前域（Vorfeld）は，1 語の場合もあれば，かなり長い句の場合もある。人間では一目瞭然のことでも，Prolog は与えられた文字列を左から右へ，逐次たくさんの構文規則を参照しながら，解析の可能性を試行錯誤していく。一つの解析が失敗すると，別の可能性を求めてバックトラックを繰り返す。それが終わってからしか移動の帰着点が決められないというのでは，副文（つまり派生形）用にもう一つ別の構文規則を導入するのと結果的に同じであり，基底形を設定する意味さえなくなる。

一方，基底形を副文の SOV 構造，派生形を主文の SVO 構造とした場合，定形の確定は，自動的に文の末尾と決まる前者に比べると少し複雑なプロセ

スが必要である。しかし次の移動規則の方は簡単で，いったん定形が確定されれば，それを自動的に末尾に移動すればよいのである。定形の確定に関しても，想像するほど効率が落ちることはない。これは，定形が，文の比較的前に位置すること，および句のかたちに拡大されることはなく，常に1語であるという特性から来ている。Prologの動作は，左から右へと逐次的に進む。つまり前にある要素ほど解析の処理は早く実行される。また常に1語であることで，複雑な句構造規則を試行錯誤することがない。このため文の前域（定形の前の文成分）が確定された段階で，次の要素が定形であることが自動的に決定できるのである。

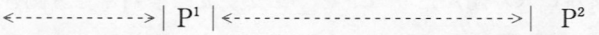
以上のように機械処理の面からは，SOV構造を基底形とするのが断然有利である。これを支持する補足的な事例として，もう一点，分離動詞の例をあげておこう。分離動詞は，通常の辞書の記載と同様，1語の不定形で辞書部門に登録されている。したがって機械が辞書引きする場合には，分離した部分をどこかで接着するルールを組み込んでおく必要があるが，基底形をSOVとした場合，分離動詞は文末で必ず隣接しているので，このルールが極めて単純に記述できる。一方，基底形がSVOの場合は，分離前綴を定形の前に移動させるルールが別途必要になる。さらにこの移動は主文の定形の前に前綴を移動させるわけで，現実の文には有り得ない構造を設定することになり，解析システムをいたずらに複雑にするだけのように思われる。

4.3. 自由語順の処理と構文規則

ドイツ語は，軸となる定形の位置は固定されているが，その他の文成分の語順はかなり自由である。この自由語順は，二つのレベルに分けて考えることができる。一つは，(3a)のように，文頭，つまり文の前域（Vorfeld）に，主語以外にもさまざまな文成分が置かれ得る，というレベル。もう一つは，(3b)のように，定形の後，いわゆる中域（Mittelfeld）において，文成分が比較的自由な配置をとる，というレベルである。

(3a) *Er* hat gestern ein schönes Mädchen getroffen.
Gestern hat er ein schönes Mädchen getroffen.
Ein schönes Mädchen hat er gestern getroffen.

(3b) Mein Freund hat *im Urlaub diesen Roman* gelesen.
Mein Freund hat *diesen Roman im Urlaub* gelesen.



このような自由語順を標準 Prolog で扱うのは，かなり難しい。Prolog の動作原理が，順序固定のリスト処理を前提にしているからである。付録1に掲げたプログラムは，その一つの解決策である[4]。ここでは，コードをなるべく簡潔にする意味で，(3a)の場合，つまり文の前域における自由語順だけを扱っている。概略を述べると，あまりエレガントな方法とは言えないが，前域が名詞句の場合と副詞句の場合をそれぞれ別の節として導入し（22行目～25行目の述語 topic/5），処理している。

プログラムは，構文規則（1行目～122行目），（前稿で扱った）形態素規則（126行目～444行目），辞書（448行目～537行目）およびリスト処理述語（541行目～555行目）の各モジュールからなる。なおこのプログラムでは，入出力のユーザインターフェイス部分，つまり，キーボードからドイツ文を入力し，解析結果を表示する部分を省略している。付録2は，構文解析の実行例である。以下，自由語順の処理を中心に，プログラムの説明をしておこう。

まず，解析の始発は述語 sentence/3（5行目）である。ここには掲載していないが，あらかじめ入出力モジュールで，入力文が Prolog のリストに変換され，この解析モジュールに引き渡される。述語 sentence/3 の第1引数 List がこれである。第2引数 Tree には構文解析結果が，第3引数 Jpn には日本語訳が返される。Tree は，最終的に入出力モジュールにおいて，

インデントされたかたちでプリティプリントされる。述語 sentence/3 の節の右辺，6 行目の s_dec/4 は平叙文を解析する述語である。なお 7 行目の flat/2 は，プログラムの 541 行目以降にあるリスト処理述語の一つで，

[a, [b], [[c, d], [e, [f, g], h]]] → [a, b, c, d, e, f, g, h]

のように，複雑に入り組んだリストを，フラットに整列させる働きをする。主に日本語訳を整形する部分に用いられている。

本試みでの文の解析手順を，もう一度確認しておこう。構文規則は，4.1, 4.2 で検討したように，階層的句構造，基底形 SOV 構造によっている。まず最初に文（平叙文）の定形の確定が試みられる。これが確定されると，文の前域は，これより前，つまり左ということが自動的に決定される。平叙文は派生形なので，この確定された定形を基底形（つまり副文）の位置，すなわち文末に移動させる。そして元の定形の位置の直後から文末（移動された定形の位置）までの文字列が，あらためて構文解析される。

この作業をおこなうのが，9 行目の s_dec/4 の節である。この動作を見てみよう。節の右辺の最初（10 行目）の append/3 で，まず文の定形の確定がおこなわれる。第 2 引数のリスト［V|Mittel］の V が定形の候補であり，これと次の述語 v_finit/11 の第 1 引数とのユニフィケーションが成功することにより，定形の確定がおこなわれる。節の右辺の三つめ（12 行目）の topic/5 は，定形の前の部分，つまり文の前域を表している。この部分は 22〜25 行目の topic/5 の節により解析される。ここの記述に見られるように，ここに来られるのは np（名詞句）か adv（副詞句）かである。s_dec/4 の節の右辺の四つめ，13 行目の append/3 は，定形を文末に移動するもので，本試みにおける移動規則の核心はこの部分にある。移動された結果は，第 3 引数 Prop に入り，これが次の述語 s_prop/9（14 行目）の第 1 引数に引き継がれる。この s_prop/9 は，定形文末の基底形の構文解析をおこなうト

ップエンドの節で，実質的な構文解析はここからスタートする。

27 行目から 33 行目は述語 s_prop/9 の定義である。これは二つの節からなり，最初の節（27〜29 行目）は，1 格主語のない述部のみからなる場合の記述，第二の節（30〜33 行目）は，1 格主語と述部からなる場合の記述である。topic/5，すなわち文の前域が，名詞句 1 格主語の場合は，最初の節にユニファイし，それ以外の場合は，第二の節にユニファイする。以下トップダウンのプロセスで，predicate/8 → vp/9 → vp 0/10 → vp 1/10 → vp 2/11 … と順に解析が実行される。vp 2/11 は述語の処理で，第一の節（64〜66 行目）は，述語が定形のみの場合，第二の節（67〜77 行目）は，述語が分離動詞の場合で，文末の二つの要素を接着させて辞書引きさせるプロセスが記述されている。第三の節（78〜81 行目）と第四の節（82〜84 行目）は助動詞構文で，前者は，haben 支配の現在完了形，後者は，sein 支配の現在完了形を扱うためのものである。ここでは枠構造処理のデモという意味で，これだけの規則にとどめているが，他の助動詞構文や受動完了形など複合的助動詞構文を含めて網羅的に処理させるには，もちろん規則の拡充が必要である。

92 行目からの述語 obj/8 以下の節は，名詞句および副詞句といった文成分を解析するための規則である。obj/8 および adv/5 の最後の節に，’△’ というアトム（定数）が見られるが，これは空の要素を意味するもので，解析対象文に，目的語や補足語（Ergänzung）といった文の必須成分が欠けている場合，構文解析結果において，そこが空になっているということを表すためのものである。これはまた，主語以外の文の必須成分が，文の前域に置かれた文の場合にも導入され，前域要素と空の要素との照応関係が構文解析結果に明示される（付録 2 の解析例 1c，2c 参照）。もちろん各要素の統語特性によっては，こうした照応関係が成り立たない場合もあるが，その場合は非文として，解析自体が失敗するようにしている。

126 行目以降 444 行目までの長いコードは，前稿で扱った形態素解析規則の一部である。まず，130 行目から 139 行目までに不規則動詞の変化形があ

げられている。140 行目から 195 行目までが規則動詞の現在人称変化規則である。動詞に限らず不規則変化形は，規則変化の特例として，辞書部門ではなく，この形態素部門で処理される。したがって辞書部門の記載項目は，基本形（動詞なら不定形，名詞なら 1 格形）のみに限られる。197 行目から 216 行目は過去分詞の規則である。230 行目から 317 行目までは，名詞の格変化規則である。名詞は，この規則を経て最終的に辞書項目にアクセスされるわけであるが，ここに該当する単語がない場合，つまり辞書にその単語が登録されていない場合，279 行目から 293 行目の述語 n_unknown/4 とユニファイし，日本語訳において，これを訳出しないで，もとの語形のまま表示するようにしている（付録 2 の解析例 2a，2b，2c 参照）。名詞の場合，語彙数は固有名詞などを含めると無限と言ってもよいので，この処理が必要である。322 行目から 444 行目までは，付加語形容詞の格変化規則である。以上，それぞれの形態素規則の動作については，前稿(1)第 2 章で扱っているので，ここではとくにふれない。最後に，448 行目以降は辞書部門である。ここでは解析の例示に必要な最小限の単語だけをあげているが，辞書の形式は，概略これで理解できると思われる。

（つづく）

〔注〕

1) 野村（1987），Hosaka（1988）参照。
2) 田中（1989）参照。
3) 重藤（1983）参照。
4) Arity Prolog Versin 5.1（NEC PC-9800 Series）による。

〔参考文献〕

重藤　実（1983）：「ドイツ語の語順と類型論」（ドイツ文学 71，S. 25–35）
ピーター・セルズ　郡司隆男他訳（1988）：現代の文法理論　産業図書
田中穂積（1989）：自然言語解析の基礎　産業図書
野村泰幸（1987）：「ドイツ語生成統語論の課題」（ドイツ文学 79，S. 55–67）
淵　一博監修（1986）：自然言語の基礎理論　共立出版

Hosaka, Y. (1988) : Einige Gründe für die Konfigurationalität im Deutschen (エネルゲイア 14, S. 70–82)

Drach, E. (1937, 1963[4]) : Grundgedanken der deutschen Satzlehre. Darmstadt.

Duden-Grammatik (1984) : Drosdowski, G. (Hg.) Die Grammatik. 4., neubearb. Aufl. (= Der Duden Bd. 4). Mannheim.

Engel, U. (1988) : Deutsche Grammatik. Heidelberg.

Erben, J. (1968, 1983[12]) : Deutsche Grammatik. Ein Leitfaden. Frankfurt a. M. / Hamburg.

Helbig, G. / Buscha, J. (1972, 1989[12]) : Deutsche Grammatik. Leipzig.

Schulz, D. / Griesbach, H. (1960, 1982[11]) : Grammatik der deutschen Sprache. Ismaning.

〔付　録　1〕

```
% ----------------
% SYNTACTIC RULES
% ----------------

sentence(List,Tree,Jpn) :-
    s_dec(List,[],Tree,Z),
    flat([Z,。],Jpn).

s_dec(A,[],s(Tree1,Tree2),[Jpn1,Jpn2]) :-
    append(Vor,[V|Mittel],A),
    v_finit([V],[],Typ,Prs,Num,_,_,Syn,_,_,_),
    topic(Vor,[],Case,Tree1,Jpn1),
    append(Mittel,[V],Prop),
    s_prop(Prop,[],Typ,Prs,Num,Case,Syn,Tree2,JList),
    (flat(JList,JListF),
    append(P,[' △ ',Q|R],JListF),
    flat([Q,P,R],Jpn2),
    !
    ; flat(JList,Jpn2)
    ).

topic(A,B,Case,Tree,Jpn) :-
    np(A,B,_,_,_,Case,_,Tree,Jpn).
topic(A,B,_,Tree,Jpn) :-
    adv(A,B,Syn,Tree,Jpn).

s_prop(A,[],Typ,Prs,Num,nom,Syn,Tree,[は,Jpn]) :-
    predicate(A,[],Typ,Prs,Num,Syn,Tree,Jpn),
    !.
s_prop(A,B,Typ,Prs,Num,Case,Syn,s0(T1,T2),[J1,は,J2]) :-
    Case ¥== nom,
    np(A,X,Prs,Num,_,nom,_,T1,J1),
    predicate(X,B,Typ,Prs,Num,Syn,T2,J2).

predicate(A,B,Typ,Prs,Num,Syn,vp(T1,T2),Jpn):-
    vp(A,B,Typ,Prs,Num,_,Syn,[T1,T2],Jpn),
    !.
predicate(A,B,Typ,Prs,Num,Syn,vp(T1,T2,T3),[J1,J2]):-
    adv(A,X,Syn1,T1,J1),
    Tree1 ¥== adv(' △ '),
    vp(X,B,Typ,Prs,Num,_,Syn,[T2|T3],J2).
```

```
43:  vp(A,B,i,Prs,Num,Tns,Syn,[Tree],Jpn) :-
44:      vp0(A,B,i,Prs,Num,Tns,Syn,Sem,Tree,Jpn),
45:      !.
46:  vp(A,B,Typ,Prs,Num,Tns,Syn,[T1,T2],[J1,J2]) :-
47:      obj(A,X,Typ,_,_,_,T1,J1),
48:      vp0(X,B,Typ,Prs,Num,Tns,Syn,Sem,T2,J2).
49:
50:
51:  vp0(A,B,Typ,Prs,Num,ps,Syn,Sem,T,[J1,J2,J3,J4]) :-
52:      vp1(A,B,Typ,Prs,Num,ps,Syn,Sem,T,[J1,J2,J3,J4,_,_]),
53:      !.
54:  vp0(A,B,Typ,Prs,Num,pt,Syn,Sem,T,[J1,J2,J3,J6]) :-
55:      vp1(A,B,Typ,Prs,Num,pt,Syn,Sem,T,[J1,J2,J3,_,_,J6]).
56:
57:  vp1(A,B,Typ,Prs,Num,Tns,Syn,Sem,vb(Tree),[[]|Jpn]) :-
58:      vp2(A,B,Typ,Prs,Num,Tns,fr,_,Sem,Tree,Jpn),
59:      !.
60:  vp1(A,B,Typ,Prs,Num,Tns,Syn,Sem,vb(T1,T2),[J1,J2|J3]) :-
61:      adv(A,X,Syn,T1,J2),
62:      vp2(X,B,Typ,Prs,Num,Tns,_,Syn,Sem,T2,[J1|J3]).
63:
64:  vp2(A,B,Typ,Prs,Num,Tns,Syn,Sem,Inf,Tree,Jpn) :-
65:      v_finit(A,B,Typ,Prs,Num,Tns,Syn,Sem,Inf,Tree,Jpn),
66:      !.
67:  vp2([A,X|B],B,Typ,Prs,Num,Tns,Syn,Sem,_,v(TV),Jpn) :-
68:      v(X,_,_,Prs,Num,Tns,_,_,Inf,_),
69:      name(A,ACode),
70:      name(Inf,InfCode),
71:      append(ACode,InfCode,FinCode),
72:      name(Lex,FinCode),
73:      v(Lex,inf,Typ,_,Syn,Sem,Jpn),
74:      name(X,XCode),
75:      append(ACode,XCode,TVCode),
76:      name(TV,TVCode),
77:      !.
78:  vp2(A,B,Typ,Prs,Num,pt,Syn,Sem,_,prd(T1,T2),Jpn) :-
79:      v_pp(A,X,Typ,h,pp,Syn,Sem,_,T1,Jpn),
80:      v_finit(X,B,_,Prs,Num,ps,_,_,haben,T2,_),
81:      !.
82:  vp2(A,B,Typ,Prs,Num,pt,Syn,Sem,_,prd(T1,T2),Jpn) :-
83:      v_pp(A,X,Typ,s,pp,Syn,Sem,_,T1,Jpn),
84:      v_finit(X,B,_,Prs,Num,ps,_,_,sein,T2,_).
85:
86:  v_finit([A|B],B,Typ,Prs,Num,Tns,Syn,Sem,Inf,v(A),Jpn) :-
87:      v(A,Typ,Perf,Prs,Num,Tns,Syn,Sem,Inf,Jpn).
```

```
 88:
 89: v_pp([A|B],B,Typ,Perf,pp,Syn,Sem,Inf,pp(A),Jpn) :-
 90:     v_pp(A,pp,Perf,Typ,Syn,Sem,Inf,Jpn).
 91:
 92: obj(A,B,t,Num,Prs,Sem,Tree,Jpn) :-
 93:     np(A,B,Num,Prs,_,akk,Sem,Tree,Jpn).
 94: obj(A,A,t,[],[],[],akk(' △ '),[' △ ']).
 95: obj(A,B,i3,Num,Prs,Sem,Tree,Jpn) :-
 96:     np(A,B,Num,Prs,_,dat,Sem,Tree,Jpn).
 97: obj(A,A,i3,[],[],[],dat(' △ '),[' △ ']).
 98:
 99: np(A,B,Num,Prs,pn,Case,Sem,np(Case,Tree),Jpn) :-
100:     pn(A,B,Num,Prs,pn,Case,Sem,Tree,Jpn).
101: np(A,B,Num,3,Sex,Case,Sem,np(Case,T1,T2),[J1,J2]) :-
102:     det(A,X,Det,Num,Sex,Case,T1,J1),
103:     nominal(X,B,Det,Num,Sex,Case,Sem,T2,J2).
104: np(A,B,Num,3,Sex,Case,Sem,np(Case,Tree),Jpn) :-
105:     nominal(A,B,_,Num,Sex,Case,Sem,Tree,Jpn).
106:
107: nominal(A,B,_,Num,Sex,Case,Sem,Tree,Jpn) :-
108:     noun(A,B,Num,Sex,Case,Sem,Tree,Jpn).
109: nominal(A,B,Det,Num,Sex,Case,Sem,n(T1,T2),[J1,J2,J3]) :-
110:     adj_attr(A,X,Det,Sex,Case,T1,[J1,J2,_]),
111:     noun(X,B,Num,Sex,_,Sem,T2,J3).
112: nominal(A,B,_,_,_,_,_,Tree,Jpn) :-
113:     n_unknown(A,B,Tree,Jpn).
114:
115: adv(A,B,Syn,adv(T1,T2),[J1,J2]) :-
116:     prep(A,X,Case,Syn,T1,J2),
117:     np(X,B,_,_,_,Case,_,T2,J1),
118:     !.
119: adv([A|B],B,Syn,adv(A),[Jpn]) :-
120:     adv(A,Syn,Jpn),
121:     !.
122: adv(A,A,Syn,adv(' △ '),[]).
123:
124:
125:
126: % --------------------
127: % MORPHOLOGICAL RULES
128: % --------------------
129:
130: v(hast,t,h,sg,2,ps,fr,_,haben,[を,もってい,る,[],た]).
131: v(hat,t,h,sg,3,ps,fr,_,haben,[を,もってい,る,[],た]).
132: v(bin,i,s,sg,1,ps,fr,_,sein,[[],で,ある,[],した]).
```

```
133: v(bist,i,s,sg,2,ps,fr,_,sein,[[],で,ある,[],した]).
134: v(ist,i,s,sg,3,ps,fr,_,sein,[[],で,ある,[],した]).
135: v(sind,i,s,pl,1,ps,fr,_,sein,[[],で,ある,[],した]).
136: v(seid,i,s,pl,2,ps,fr,_,sein,[[],で,ある,[],した]).
137: v(sind,i,s,pl,3,ps,fr,_,sein,[[],で,ある,[],した]).
138: v(triffst,t,h,sg,2,ps,fr,_,treffen,[に,出会,う,わ,った]).
139: v(trifft,t,h,sg,3,ps,fr,_,treffen,[に,出会,う,わ,った]).
140: v(A,B,C,sg,3,ps,D,E,F,G) :-
141:     name(A,H),
142:     append(I,[116],H),
143:     append(I,[101,110],J),
144:     name(F,J),
145:     v(F,inf,B,C,D,E,G).
146: v(A,B,C,pl,3,ps,E,F,A,G) :-
147:     v(A,inf,B,C,E,F,G).
148: v(A,B,C,sg,1,ps,D,E,F,G) :-
149:     name(A,H),
150:     append(I,[101],H),
151:     append(I,[101,110],J),
152:     name(F,J),
153:     v(F,inf,B,C,D,E,G).
154: v(A,B,C,sg,2,ps,D,E,F,G) :-
155:     name(A,H),
156:     append(I,[115,116],H),
157:     append(I,[101,110],J),
158:     name(F,J),
159:     v(F,inf,B,C,D,E,G).
160: v(A,B,C,pl,1,ps,D,E,A,F) :-
161:     v(A,inf,B,C,D,E,F).
162: v(A,B,C,pl,2,ps,D,E,F,G) :-
163:     name(A,H),
164:     append(I,[116],H),
165:     append(I,[101,110],J),
166:     name(F,J),
167:     v(F,inf,B,C,D,E,G).
168: v(A,B,C,sg,1,pt,D,E,F,G) :-
169:     v(A,pt,B,C,D,E,F,G).
170: v(A,B,C,sg,2,pt,D,E,F,G) :-
171:     name(A,H),
172:     append(I,[115,116],H),
173:     name(J,I),
174:     v(J,pt,B,C,D,E,F,G).
175: v(A,B,C,sg,3,pt,D,E,F,G) :-
176:     v(A,pt,B,C,D,E,F,G).
177: v(A,B,C,pl,1,pt,D,E,F,G) :-
```

```
178:      name(A,H),
179:      ( append(I,[101,110],H)
180:      ; append(I,[110],H)
181:      ),
182:      name(J,I),
183:      v(J,pt,B,C,D,E,F,G).
184:  v(A,B,C,pl,2,pt,D,E,F,G) :-
185:      name(A,H),
186:      append(I,[116],H),
187:      name(J,I),
188:      v(J,pt,B,C,D,E,F,G).
189:  v(A,B,C,pl,3,pt,D,E,F,G) :-
190:      name(A,H),
191:      ( append(I,[101,110],H)
192:      ; append(I,[110],H)
193:      ),
194:      name(J,I),
195:      v(J,pt,B,C,D,E,F,G).
196:
197:  v_pp(gegangen,pp,s,i,fr,_,gehen,[[],行,く,か,った]).
198:  v_pp(gekommen,pp,s,i,fr,_,kommen,[[],来,る,[],た]).
199:  v_pp(getroffen,pp,h,t,fr,_,treffen,[に,出会,う,わ,った]).
200:  v_pp(A,pp,B,C,D,E,F,G) :-
201:      name(A,H),
202:      append([103,101],I,H),
203:      append(J,[116],I),
204:      append(J,[101,110],K),
205:      name(F,K),
206:      v(F,inf,C,B,D,E,G).
207:  v_pp(A,pp,B,C,D,E,F,G) :-
208:      name(A,H),
209:      append(I,[103,101|J],H),
210:      I ¥== [],
211:      name(K,[103,101|J]),
212:      v_pp(K,pp,L,M,N,O,P,Q),
213:      name(P,R),
214:      append(I,R,S),
215:      name(F,S),
216:      v(F,inf,C,B,D,E,G).
217:
218:  prep([A|B],B,Case,D,prep(A),E) :-
219:      prep(A,Case,D,E).
220:
221:  det([A|B],B,C,sg,m,D,det(A),E) :-
222:      det(A,C,m,D,E).
```

```
223:  det([A|B],B,C,sg,n,D,det(A),E) :-
224:      det(A,C,n,D,E).
225:  det([A|B],B,C,sg,f,D,det(A),E) :-
226:      det(A,C,f,D,E).
227:  det([A|B],B,C,pl,pl,D,det(A),E) :-
228:      det(A,C,pl,D,E).
229:
230:  noun([A|B],B,sg,m,C,D,n(A),E) :-
231:      noun(A,m,C,D,E).
232:  noun([A|B],B,sg,n,C,D,n(A),E) :-
233:      noun(A,n,C,D,E).
234:  noun([A|B],B,sg,f,C,D,n(A),E) :-
235:      noun(A,f,C,D,E).
236:  noun([A|B],B,pl,pl,C,D,n(A),E) :-
237:      noun(A,pl,C,D,E).
238:
239:  noun(A,pl,nom,B,C) :-
240:      n(D,m,[E,A],B,C).
241:  noun(A,pl,akk,B,C) :-
242:      n(D,m,[E,A],B,C).
243:  noun(A,pl,nom,B,C) :-
244:      n(D,f,[A],B,C).
245:  noun(A,pl,akk,B,C) :-
246:      n(D,f,[A],B,C).
247:  noun(A,pl,nom,B,C) :-
248:      n(D,n,[E,A],B,C).
249:  noun(A,pl,akk,B,C) :-
250:      n(D,n,[E,A],B,C).
251:  noun(A,pl,dat,B,C) :-
252:      ( name(A,D),
253:        append(E,[110],D),
254:        name(F,E),
255:        n(G,m,[H,F],B,C)
256:      ; n(G,m,[H,A],B,C)
257:      ).
258:  noun(A,pl,dat,B,C) :-
259:      ( name(A,D),
260:        append(E,[110],D),
261:        name(F,E),
262:        n(G,f,[F],B,C)
263:      ; n(G,f,[A],B,C)
264:      ).
265:  noun(A,pl,dat,B,C) :-
266:      ( name(A,D),
267:        append(E,[110],D),
```

```
268:          name(F,E),
269:          n(G,n,[H,F],B,C)
270:        ; n(G,n,[H,A],B,C)
271:        ).
272:  noun(A,B,nom,C,D) :-
273:        n(A,B,[[E],F],C,D).
274:  noun(A,B,dat,C,D) :-
275:        n(A,B,[[E],F],C,D),
276:        name(E,G),
277:        ¥+ append(H,[110],G).
278:  noun(A,m,dat,B,C) :-
279:        n(D,m,[[A],E],B,C),
280:        name(A,F),
281:        append(G,[110],F).
282:  noun(A,B,akk,C,D) :-
283:        n(A,B,[[E],F],C,D),
284:        name(E,G),
285:        ¥+ append(H,[110],G).
286:  noun(A,m,akk,B,C) :-
287:        n(D,m,[[A],E],B,C),
288:        name(A,F),
289:        append(G,[110],F).
290:  noun(A,B,nom,C,D) :-
291:        n(A,B,[[E,F],G],C,D).
292:  noun(A,B,dat,C,D) :-
293:        n(A,B,[[E,F],G],C,D).
294:  noun(A,B,akk,C,D) :-
295:        n(A,B,[[E,F],G],C,D).
296:  noun(A,f,nom,B,C) :-
297:        n(A,f,[D],B,C).
298:  noun(A,f,dat,B,C) :-
299:        n(A,f,[D],B,C).
300:  noun(A,f,akk,B,C) :-
301:        n(A,f,[D],B,C).
302:
303:  n_unknown([A|B],B,n(A),A) :-
304:        name(A,[C|D]),
305:        C > 64,
306:        C < 91,
307:        ¥+ noun(A,E,F,G,H),
308:        ¥+ pron(A,I,J,K,L,M),
309:        ¥+ det(A,N,O,P),
310:        Q is C + 32,
311:        name(R,[Q|D]),
312:        ¥+ pron(R,S,T,U,V,W),
```

```
313:      ¥+ adv(R,X,Y),
314:      ¥+ adj(R,Z,A1),
315:      ¥+ v(R,B1,C1,D1,E1,F1,G1),
316:      ¥+ det(R,H1,I1,J1,K1),
317:      !.
318:
319:  pn([A|B],B,Num,Prs,pn,Case,Def,pron(A),Jpn) :-
320:      pron(A,Num,Prs,Case,Def,Jpn).
321:
322:  adj_attr([A|B],B,def,m,nom,adj(A),C) :-
323:      name(A,D),
324:      append(E,[101],D),
325:      name(F,E),
326:      adj(F,G,C).
327:  adj_attr([A|B],B,def,m,dat,adj(A),C) :-
328:      name(A,D),
329:      append(E,[101,110],D),
330:      name(F,E),
331:      adj(F,G,C).
332:  adj_attr([A|B],B,def,m,akk,adj(A),C) :-
333:      name(A,D),
334:      append(E,[101,110],D),
335:      name(F,E),
336:      adj(F,G,C).
337:  adj_attr([A|B],B,def,f,nom,adj(A),C) :-
338:      name(A,D),
339:      append(E,[101],D),
340:      name(F,E),
341:      adj(F,G,C).
342:  adj_attr([A|B],B,def,f,dat,adj(A),C) :-
343:      name(A,D),
344:      append(E,[101,110],D),
345:      name(F,E),
346:      adj(F,G,C).
347:  adj_attr([A|B],B,def,f,akk,adj(A),C) :-
348:      name(A,D),
349:      append(E,[101],D),
350:      name(F,E),
351:      adj(F,G,C).
352:  adj_attr([A|B],B,def,n,nom,adj(A),C) :-
353:      name(A,D),
354:      append(E,[101],D),
355:      name(F,E),
356:      adj(F,G,C).
357:  adj_attr([A|B],B,def,n,dat,adj(A),C) :-
```

```
358:      name(A,D),
359:      append(E,[101,110],D),
360:      name(F,E),
361:      adj(F,G,C).
362:  adj_attr([A|B],B,def,n,akk,adj(A),C) :-
363:      name(A,D),
364:      append(E,[101],D),
365:      name(F,E),
366:      adj(F,G,C).
367:  adj_attr([A|B],B,def,pl,nom,adj(A),C) :-
368:      name(A,D),
369:      append(E,[101,110],D),
370:      name(F,E),
371:      adj(F,G,C).
372:  adj_attr([A|B],B,def,pl,dat,adj(A),C) :-
373:      name(A,D),
374:      append(E,[101,110],D),
375:      name(F,E),
376:      adj(F,G,C).
377:  adj_attr([A|B],B,def,pl,akk,adj(A),C) :-
378:      name(A,D),
379:      append(E,[101,110],D),
380:      name(F,E),
381:      adj(F,G,C).
382:  adj_attr([A|B],B,indef,m,nom,adj(A),C) :-
383:      name(A,D),
384:      append(E,[101,114],D),
385:      E ¥== [],
386:      name(F,E),
387:      adj(F,G,C).
388:  adj_attr([A|B],B,indef,m,dat,adj(A),C) :-
389:      name(A,D),
390:      append(E,[101,110],D),
391:      name(F,E),
392:      adj(F,G,C).
393:  adj_attr([A|B],B,indef,m,akk,adj(A),C) :-
394:      name(A,D),
395:      append(E,[101,110],D),
396:      name(F,E),
397:      adj(F,G,C).
398:  adj_attr([A|B],B,indef,f,nom,adj(A),C) :-
399:      name(A,D),
400:      append(E,[101],D),
401:      name(F,E),
402:      adj(F,G,C).
```

```
403:  adj_attr([A|B],B,indef,f,dat,adj(A),C) :-
404:      name(A,D),
405:      append(E,[101,110],D),
406:      name(F,E),
407:      adj(F,G,C).
408:  adj_attr([A|B],B,indef,f,akk,adj(A),C) :-
409:      name(A,D),
410:      append(E,[101],D),
411:      name(F,E),
412:      adj(F,G,C).
413:  adj_attr([A|B],B,indef,n,nom,adj(A),C) :-
414:      name(A,D),
415:      append(E,[101,115],D),
416:      E ¥== [],
417:      name(F,E),
418:      adj(F,G,C).
419:  adj_attr([A|B],B,indef,n,dat,adj(A),C) :-
420:      name(A,D),
421:      append(E,[101,110],D),
422:      name(F,E),
423:      adj(F,G,C).
424:  adj_attr([A|B],B,indef,n,akk,adj(A),C) :-
425:      name(A,D),
426:      append(E,[101,115],D),
427:      E ¥== [],
428:      name(F,E),
429:      adj(F,G,C).
430:  adj_attr([A|B],B,indef,pl,nom,adj(A),C) :-
431:      name(A,D),
432:      append(E,[101,110],D),
433:      name(F,E),
434:      adj(F,G,C).
435:  adj_attr([A|B],B,indef,pl,dat,adj(A),C) :-
436:      name(A,D),
437:      append(E,[101,110],D),
438:      name(F,E),
439:      adj(F,G,C).
440:  adj_attr([A|B],B,indef,pl,akk,adj(A),C) :-
441:      name(A,D),
442:      append(E,[101,110],D),
443:      name(F,E),
444:      adj(F,G,C).
445:
446:
447:
```

```
448:  % --------
449:  % LEXICON
450:  % --------
451:
452:  v(ankommen,inf,i,s,in,place,[[],到着,する,し,した]).
453:  v(danken,inf,i3,h,fr,_,[に,感謝,する,し,した]).
454:  v(gehen,inf,i,s,fr,dir,[[],行,く,か,った]).
455:  v(haben,inf,t,h,fr,_,[を,もってい,る,[],た]).
456:  v(kommen,inf,i,s,fr,place,[[],来,る,[],た]).
457:  v(lieben,inf,t,h,fr,_,[を,愛して,いる,い,いた]).
458:  v(sein,inf,il,s,fr,_,[[],で,す,[],した]).
459:  v(suchen,inf,t,h,fr,_,[を,探,す,さ,した]).
460:  v(treffen,inf,t,h,fr,_,[に,出会,う,わ,った]).
461:  v(wohnen,inf,i,h,syn,place,[[],住んでい,る,[],た]).
462:
463:  adv(gestern,time1,昨日).
464:  adv(heute,time1,今日).
465:  adv(manchmal,time2,ときどき).
466:  adv(morgen,time1,明日).
467:  adv(oft,time2,しばしば).
468:
469:  prep(in,dat,place,に).
470:  prep(in,akk,goal,に).
471:  prep(mit,dat,A,といっしょに).
472:  prep(nach,dat,goal,へ).
473:  prep(von,dat,A,から).
474:
475:  det(der,def,m,nom,[その]).
476:  det(dem,def,m,dat,[その]).
477:  det(den,def,m,akk,[その]).
478:  det(die,def,f,nom,[その]).
479:  det(der,def,f,dat,[その]).
480:  det(die,def,f,akk,[その]).
481:  det(das,def,n,nom,[その]).
482:  det(dem,def,n,dat,[その]).
483:  det(das,def,n,akk,[その]).
484:  det(die,def,pl,nom,[その]).
485:  det(den,def,pl,dat,[その]).
486:  det(die,def,pl,akk,[その]).
487:  det(ein,indef,m,nom,[]).
488:  det(einem,indef,m,dat,[]).
489:  det(einen,indef,m,akk,[]).
490:  det(eine,indef,f,nom,[]).
491:  det(einer,indef,f,dat,[]).
492:  det(eine,indef,f,akk,[]).
```

```
493:  det(ein,indef,n,nom,[]).
494:  det(einem,indef,n,dat,[]).
495:  det(ein,indef,n,akk,[]).
496:
497:  n('Frau',f,['Frauen'],human,女性).
498:  n('Kind',n,[['Kindes'],'Kinder'],human,子供).
499:  n('Lehrer',m,[['Lehrers'],'Lehrer'],human,先生).
500:  n('Lehrerin',f,['Lehrerinnen'],human,女教師).
501:  n('Mann',m,[['Manns','Mannes'],'M|nner'],human,男).
502:  n('Mutter',f,['Mutter'],human,母).
503:  n('M|dchen',n,[['M|dchens'],'M|dchen'],human,少女).
504:  n('Student',m,[['Studenten'],'Studenten'],human,学生).
505:  n('Studentin',f,['Studentinnen'],human,女学生).
506:  n('Vater',m,[['Vaters'],'V|ter'],human,父).
507:
508:  pron(ich,sg,1,nom,per,私).
509:  pron(mir,sg,1,dat,per,私).
510:  pron(mich,sg,1,akk,per,私).
511:  pron(du,sg,2,nom,per,君).
512:  pron(dir,sg,2,dat,per,君).
513:  pron(dich,sg,2,akk,per,君).
514:  pron(er,sg,3,nom,per,彼).
515:  pron(ihm,sg,3,dat,per,彼).
516:  pron(ihn,sg,3,akk,per,彼).
517:  pron(sie,sg,3,nom,per,彼女).
518:  pron(ihr,sg,3,dat,per,彼女).
519:  pron(sie,sg,3,akk,per,彼女).
520:  pron(es,sg,3,nom,per,それ).
521:  pron(ihm,sg,3,dat,per,それ).
522:  pron(es,sg,3,akk,per,それ).
523:  pron(wir,pl,1,nom,per,私たち).
524:  pron(uns,pl,1,dat,per,私たち).
525:  pron(uns,pl,1,akk,per,私たち).
526:  pron(ihr,pl,2,nom,per,君たち).
527:  pron(euch,pl,2,dat,per,君たち).
528:  pron(euch,pl,2,akk,per,君たち).
529:  pron(sie,pl,3,nom,per,彼ら).
530:  pron(ihnen,pl,3,dat,per,彼ら).
531:  pron(sie,pl,3,akk,per,彼ら).
532:
533:  adj('flei¯ig',adv,[勤勉,な,に]).
534:  adj(freundlich,non,[親切,な,に]).
535:  adj(klein,non,[小さ,い,く]).
536:  adj(reich,non,[豊か,な,[]]).
537:  adj('sch|n',non,[美しい,[],[]]).
```

```
538:
539:
540:
541: % ------------------------------
542: % PREDICATES FOR LIST PROCESSING
543: % ------------------------------
544:
545: append([],A,A).
546: append([A|B],C,[A|D]) :-
547:     append(B,C,D).
548:
549: flat([],[]).
550: flat([A|B],C) :-
551:     flat(A,D),
552:     flat(B,E),
553:     append(D,E,C),
554:     !.
555: flat(A,[A]).
```

〔付　録　2〕

【解析例 1a】

```
?- run.
独文を入力して下さい

Er hat gestern ein schönes Mädchen getroffen.


構文解析
s(
    np( nom,pron(er)),
    vp( adv(gestern),
        np(akk,det(eln),n(adj(schönes),n(Mädchen))),
        vb(prd(pp(getroffen),v(hat)))
    )
)

日本語訳
[彼は昨日美しい少女に出会った。]

yes
```

【解析例 1b】

```
?- run.
独文を入力して下さい

Gestern hat er ein schönes Mädchen getroffen.


構文解析
s(
    adv(gestern),
    s0( np( nom,pron(er)),
        vp( np(akk,det(ein),n(adj(schönes),n(Mädchen))),
            vb(prd(pp(getroffen),v(hat)))
        )
    )
)

日本語訳
[昨日彼は美しい少女に出会った。]

yes
```

【解析例 1c】

```
?- run.
独文を入力して下さい

Ein schönes Mädchen hat er gestern getroffen.


構文解析
s(
    np( akk,det(ein),n(adj(schönes),n(Mädchen))),
    s0( np(nom,pron(er)),
        vp(
            akk( △ ),
            vb(adv(gestern),prd(pp(getroffen),v(hat)))
        )
    )
)

日本語訳
［美しい少女に彼は昨日出会った。］

yes
```

【解析例 2a】

```
?- run.
独文を入力して下さい

Das Mädchen kommt heute in Berlin an.


構文解析
s(
    np( nom,det(das),n(Mädchen)),
    vp(
        adv(heute),
        vb(adv(prep(in),np(dat,n(Berlin))),v(ankommt))
    )
)
日本語訳
［その少女は今日Berlinに到着する。］

yes
```

【解析例 2b】

```
?- run.
独文を入力して下さい

Heute kommt das Mädchen in Berlin an.


構文解析
s(
    adv(heute),
    s0( np( nom,det(das),n(Mädchen)),
        vp(
            vb(adv(prep(in),np(dat,n(Berlin))),v(ankommt))
        )
    )
)

日本語訳
[今日その少女はBerlinに到着する。]

yes
```

【解析例 2c】

```
?- run.
独文を入力して下さい

In Berlin kommt das Mädchen heute an.


構文解析
s(
    adv(prep(in),np(dat,n(Berlin))),
    s0( np(nom,det(das),n(Mädchen)),
        vp(
            adv(heute),
            vb(adv( △ ),v(ankommt))
        )
    )
)

日本語訳
[Berlinにその少女は今日到着する。]

yes
```