自然言語処理 38－4
（1983. 7. 15）

文法記述用ソフトウェア GRADE*)

中村 順一（京大・工）

本報告では，機械翻訳システムのために開発した，自然言語処理のための文法記述用ソフトウェア GRADE（GRAmmar DEscriber）について述べる。GRADE には次の特長がある。
(1)部分文法の考え方を用いて文法を記述する。
(2)パターンマッチングに基づく木構造（節点に属性名と属性値の組のリストを持つ）の変換規則により個々の文法規則を記述する。
(3)辞書中にも文法規則が記述できる。
(4)形態素処理を除く翻訳のすべての過程を同一の枠組で記述できる。
(5)人間にとって書き易く，読み易い記述形式（外部形式）を用いる。

1．はじめに

自然言語処理のための文法記述用ソフトウェアを設計する際に決定すべき事柄として，次の5点が考えられる。
(1)処理対象のデータの表現方法（ex. 木構造，ネットワーク，フレーム etc）
(2)文法の一様さ（すべての規則が対等かどうか）
(3)個々の文法規則の表現方法
(4)あいまいさの扱い方（バックトラック，パラレル）
(5)辞書内容の表現方法（記述的表現，手続き的表現）
以下では，これらについて考察し，GRADE で採用した方法について述べる。なお GRADE の記述形式の詳細については，文献2 を参照のこと。

2．処理対象のデータの表現方法

データの表現方法としては，単純な木構造，属性値（属性名と属性値の組のリスト）を節点に付加した木構造（図1参照），ネットワーク表現，フレーム的表現などが考えられる。単純な木構造の表現では，自然言語処理のために必要な情報を十分に表現することはできない。この表現を用いる場合には，基本となる文法記述の形式とは異った メカニズム（たとえば，LISP 関数）が必要となり，文法の書き易さ，読み易さが損われる。フレーム的な表現を用いればデータ中に手続きを表現できるなど，非常に柔軟な表現が可能である。しかし，大規模な文法や，辞書を開発しようとした場合，フレーム的表現の長所を十分に活用できるかどうかは不明である。ネットワーク的表現は，照応関係を表現したりする上で利点がある。しかし，フレーム的表現，ネットワーク的表現は共に，あいまいさを扱う上での処理のオーバーヘッドが大きくなると予想される。これに対して，属性値を持った木構造は，自然言語の処理を行うのに必要な情報を十分に表現でき，データを操作する上での問題も少ない。そこで，GRADE では，属性値付きの木構造をデータの表現に用いることとし，文法規則中で属性値を直接操作できるようにした。

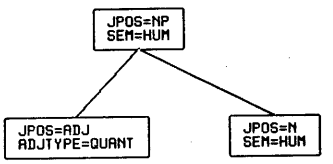


図1 属性値を付加した木構造の例

*) 本研究は国の科学技術振興調整費による「日英科学技術文献の速報システムに関する研究」の一部として行ったものである。本研究には，長尾真，辻井潤一，佐藤理史，千田裕彦（以上京大・工），小木正三，坂根嘉典（以上東洋情報システム）及びプロジェクト参加者の協力を得ている。

3. 文法の一様さと部分文法

自然言語の解析文法を考える場合，個々の文法規則が形式言語のようにすべて対等であるとするか，文法規則間に優先順位や排他性などを持たせるかを決定しなければならない。機械翻訳システムにおける文法規則にはいろいろな種類，たとえば，木構造をまとめあげる規則や，必要なマークを付加する規則，ヒューリスティックを扱う規則などがある。これらの規則はそれぞれ適用の仕方が異っているので[3]，すべての規則を一つの，一様な枠組みの中に入れてしまうことは，文法の記述の上でも，処理の効率の上でも好ましくない。そこで，GRADEでは，個々の文法規則を対等とせず，文法全体を複数の部分に分割し（以下で述べるように，これを部分文法と呼ぶ），これにより文法規則の相互関係や適用の仕方の違いを表わすこととした。

GRADEでは，文法を3つの階層に分けて記述する。これらは，部分文法ネットワーク（Sub Grammar Network, SGN）部分文法（Sub Grammar, SG），書き換え規則（Rewriting Rule, RR）である。この他に，部分文法と同じレベルに条件判定部（Conditional Brancher, CB）と，書き換え規則と同じレベルに条件判定規則（Condition check Rule, CR）がある。これらの関係を図2に示す。

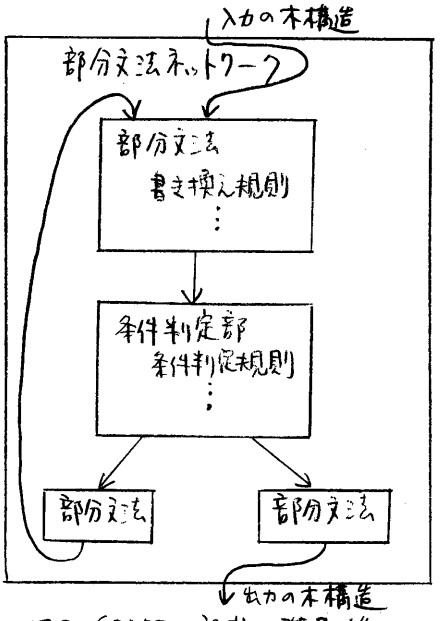


図2 GRADEの記述の階層性

書き換え規則は，後で述べるように，木構造を木構造に変換する規則であり，個々の処理，たとえば，連体詞と名詞をまとめる処理，を行うためのものである。この書き換え規則をいくつかまとめたものが部分文法である。

部分文法は，ひとまとまりの処理，たとえば，名詞句の処理，を行うためのものである。部分文法中の個々の書き換え規則には優先順位がある。そして，それらを決定的に適用する（ある規則が適用できれば，それを解除しない）か，非決定的に適用する（後で述べるように，バックトラックにより適用を解除する）かを部分文法ごとに指定できる。また，部分文法中の書き換え規則をどのように適用するのかも指定できる。適用順序を4種類に分類しており，これをORDER(1)～ORDER(4)で指定する。図3で「○」はその書き換え規則（RR）が適用されたことを示し，「→」はその部分文法の処理が終了したことを示している。たとえば，ORDER(4)は，RR_iを適用し，その結果に対してRR_jを適用し，その結果に対してRR_1～RR_nの適用を試みることを示している。この機能により，規則の適用の仕方を部分文法の目的に合せてこまかく制御することが可能である。

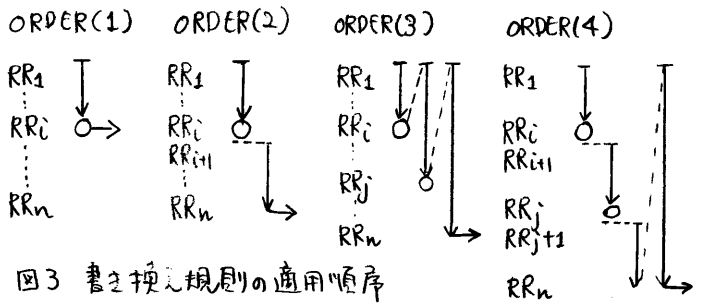


図3 書き換え規則の適用順序

条件判定部と条件判定規則は，それぞれ部分文法と書き換え規則とに同じ機能を持っているが，木構造の変換は行わず，木構造の形を検査し，次に述べる部分文法ネットワークの処理の流れを制御するために用いられる。

部分文法ネットワークは，いくつかの部分文法と条件判定部の適用順序を指定するためのもの

である。これにより，翻訳の全体の流れを指定することができる。たとえば，解析，トランスファー，生成の3つの部分文法ネットワーク*)を要素に持つ部分文法ネットワークを作り，それらを順次適用していくことや，解析のための部分文法ネットワーク中に，前処理，名詞句処理，文処理といった部分文法を作り，これらを順次適用していくこともできる。

図4にGRADEによる記述例（部分文法ネットワーク，部分文法）を示す。この例では，入力の木構造に対して，まず，PREPという部分文法が適用される。部分文法PREPでは，SU，VERB_AUXといった書き換え規則が次々に，決定的（DETERMINISTIC）に適用される。即ち，入力に対してSUが適用され，その変換結果がVERB_AUXの入力になり，次々に規則の適用が行われていく。

```
"$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ SENT.SGN $$
SENT.SGN ;
  DIRECTORY_ENTRY ;
    OWNER(NOB.TANAKA) ;
    VERSION(V02L03) ;
    LAST_UPDATE(83/5/19) ;
  ENTRY ;
    P ;
  NETWORK ;
    P : PREP.SG ;
    S : SENT.SG ;
        EXIT ;
END_SGN.SENT ;
"************************** PREP.SG
PREP.SG ;                          " APPLY
  DIRECTORY_ENTRY ;
    OWNER(NOB.TANAKA) ;
    VERSION(V02L05) ;
    LAST_UPDATE(83/6/29) ;
  SG_MODE;
    ORDER(2) ;
    DETERMINISTIC ;
  RR_IN_SG ;
    SU ;
    VERB_AUX ;          " VERB
    SV_AUX ;
    DA_AUX ;
END_SG.PREP ;
```

部分文法ネットワークの定義（部分文法の名前）

部分文法の定義（書き換え規則の名前）

図4 GRADEの記述の例

4．個々の文法規則の表現方法　2で述べたように，GRADEの扱うデータは，属性値付きの木構造であるので，個々の文法規則は，木構造を木構造に変換するものになる。構造を変換するための表現として，記述的な表現と手続き的な表現が考えられる。手続き的な表現，たとえば，LISP関数で構造変換のアルゴリズムを表現すると，非常に柔軟な表現が可能になる。しかし，規則の記述のし易さや読み易さは損われてしまう。これに対して，記述的な表現，たとえば，マッチング・パターンと生成パターンの組**)による表現では，記述のし易さ，読み易さは十分であるが，柔軟な表現ができなくなる傾向がある。たとえば，パターン・マッチングに成功した場合，そのマッチングにより生成する木構造のデータを選択したりすることができない。そこで，GRADEでは，記述的な表現，即ち，マッチング・パターンと生成パターンの組を中心とし，その間にIF-THEN-ELSE形の手続き的な表現を埋め込めるようにした。（図5参照）

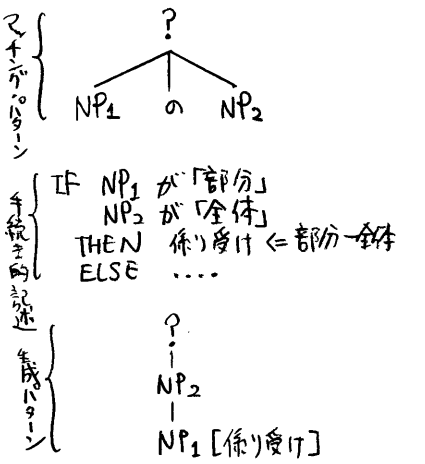


図5 木構造の変換規則

文法記述の具体的な記述形式も重要な問題である。GRADEはLISPで実現されているので，LISPのシンタックスが直接現れるような記述形式を採用することは，最も安易な方法である。しかし，文法規則は定常的に保守する必要があり，保守をする者はLISPのプログラマとは限らない。そこで，GRADEでは，文法記述者が用いる形式として，書き易く読み易いことを目標として設計した表現（外部形式）を用い，これをLISPのS式の形で表わされた内部表現に変換し，実行することにした。

GRADEの書き換え規則は，主として，4つの部分から構成されている。これらは，書き換え規則の適用方法を指定するMATCHING INSTRUCTION部，入力の木構造と

*) 部分文法ネットワークの節点として部分文法だけでなく，部分文法ネットワークも再帰的に指定できる。また，4で述べるように，書き換え規則中から部分文法ネットワークや部分文法を再帰的に呼び出すことも可能である。

**) 記述形式としてCFGの書き換え規則を例に考えると，書き換え規則（ex NP→ADJ NP）を解析の立場から見て，右辺をマッチング・パターン，左辺を生成パターンと呼ぶことにする。

構造のマッチングを行うべきパターンを記述するMATCHING CONDITION部，パターン・マッチングに成功した場合に，入力の木構造の一部の変換を行うSUBSTRUCTURE OPERATION部，変換結果の木構造を表わすパターンを記述するCREATION部である。図6に例を示す。

GRADEの書き換え規則は，入力の木構造全体に対してのみ適用されるのではなく，入力の木構造の一部分に対して適用することもできる。これは，例えば，入力の木構造のすべての動詞句に対して，一度に，同じような木構造の変換を行いたい場合に便利である。MATCHING INSTRUCTION部は，書き換え規則を入力の木構造のどの部分に対して，どのような順序で適用するかを指定する部分である。この指定は，図7に示すように4種類が可能である。たとえば，LEFT_TO_RIGHT，BOTTOM_TO_TOPの場合は，まず1の部分木が検査され，次に2の部分，次に3以下の部分木が検査される。また，繰り返し規則を適用する場合には，図8に示すように，4種類の指定ができる。たとえば，ORDER(3)では，図7の3のノードに規則が適用されたとすると，その変換を行った木に対して再び，1のノードから順に規則の適用を試みることを示している。また，次に述べるマッチング・パターンを木として扱うか，リスト（木の列）として扱うかも指定できる。

```
VP.RR ;                                  " PICK UP A UNIT 'VP' "
  MATCHING_INSTRUCTION ;
    L_TO_R ;
    ORDER(1);
    TREE;
  MATCHING_CONDITION ;
    %( ( ? $1 CONJ IADVPS VP? $2 ) ) ;
        VP?    : ANY ;
        CONJ   : OPTIONAL( %( CONJ1 ) ) ;
        IADVPS : ANY( %( COND ) | %( TEIJI ) ) ;
  SUBSTRUCTURE_OPERATION ;     " PARSE A UNIT 'VP' "
    ●VP <= CALL-SGN( VP.SGN  %( VP? ) LIST ) ;
  CREATION ;
  IF ●VP = NIL ;
         THEN 'RULE_FAIL ;
         ELSE  %( ( ? (S CONJ IADVPS ●VP ) ) ) ;
  END_IF ;
END_RR.VP ;
```

図6 GRADEの書き換え規則の例

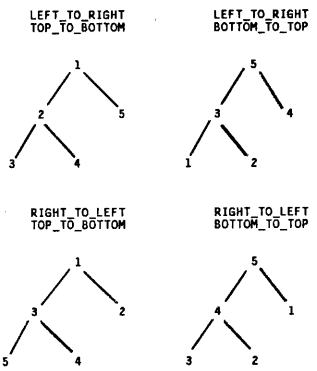


図7 書き換え規則の適用順序(1)

MATCHING CONDITION部では，入力の木構造とマッチングするための条件を，木構造と各ノードの属性値により指定する。木構造としては，任意の長さの木構造（ANY），順序任意の木構造，木構造の否定条件なども指定できる。属性値は，属性値と定数の比較だけでなく，属性値どうしの比較や変数との比較も可能である。図5では，CONJがあってもなくてもよく，その後にCONDまたはTEIJIが任意個ならび，その後にVPが任意個あってよいというパターンを表わしている。

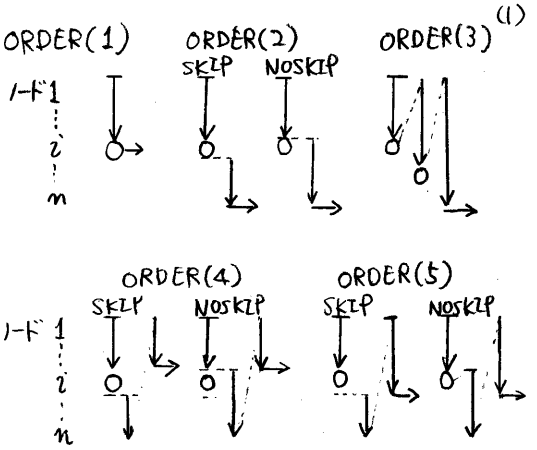


図8 書き換え規則の適用順序(2)

SUBSTRUCTURE OPERATION部では，マッチングした構造の部分構造に対する操作を指定する。ここでは，変数への値の代入，他の部分文法ネットワークや部分文法の再帰的呼出し，後で述べる辞書規則の適用，LISP関数の呼出しができる。また，IF-THEN-ELSE形の手続き的な記述もできる。図5では，CALL-SGNの部分で，VPの列をVPという部分文法ネットワークに渡して処理させることを示している。

CREATION部では，生成する構造（変換結果）を指定する。ここでもIF-THEN-ELSE形の記述が可能である。また，各ノードに属性値を付加することもできる。更に，規則の適用を強制的に失敗させることもできる。図5では，VPがまとまっていなければ規則の適用を失敗させ，まとまっていれば，Sの下にCONJ，COND又はTEIJIの列，VPを並べることを示している。

5. あいまいさの扱い方

自然言語処理において，あいまいさをどのようにして扱うかは，避けられない問題である。あいまいな解釈を得る方法として，バックトラックによるものとパラレルに処理を行うものとが考えられる。どちらの方法でもすべての可能な解釈を得ようとした場合には，同一の結果が得られるので，文法記述者から見た場合に差はない。しかし，機械翻訳システムを考えた場合，1つの文に対して，何通りもの翻訳結果を出すことは，言語現象を扱う研究の上では興味深いが，工学的立場からは問題がある。即ち，妥当な翻訳結果が最初に得られることが望ましい。また，多量の翻訳を行い，post editが可能な状況では，各文に対して唯一の訳だけを出力した方が適切であろう。この点と，実行速度，記憶容量を考え，バックトラックによりあいまいさを扱うことにした。

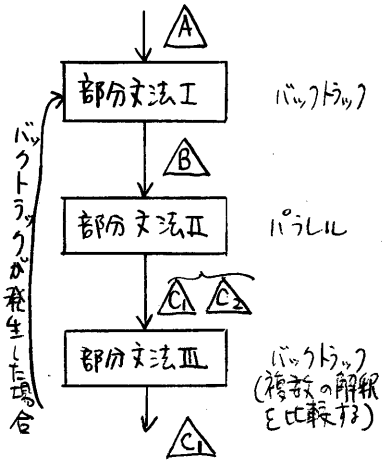


図9 あいまいさの扱い方

GRADEでは，バックトラックによりあいまいさを扱っているが，同時に複数の解釈を得て，それらの相互の比較を行いたい場合もある。たとえば，名詞句の処理で，可能な解釈をすべておしておき，動詞との関係を調べる場合に，その中から適切なものを選択するといった処理を行いたい場合がある。そこで，GRADEでは，部分文法単位でバックトラックの制御を行い，文法記述者が指定した場合は，その部分文法だけで必要なバックトラックを行ってすべての解釈が得られるようにした。（図9参照）

6. 辞書内容の表現方法

大規模な機械翻訳システムにおいては，文法記述だけでなく，辞書内容をどのように表現するかも問題である。[3)4)] 辞書内容の表現方法としては，記述的なものと手続き的なものが考えられる。ここでいう記述的な表現とは，辞書内容をパラメータにより表現し，これを文法規則が解釈することにより処理を行うことを意味している。たとえば，品詞を細分類したりすることがこれに相当する。この方法では，辞書を修正することは，パラメータの値を変更することを意味している。新たな言語現象を扱うために新しいパラメータやパラメータの値を追加した場合には，文法規則自体を修正しなければならないので，柔軟性に乏しいと考えられる。手続き的な表現とは，辞書中にその単語を扱うため専用の文法規則を書き込めるようにすることである。この方法では，辞書内容と文法規則本体との差が少なくなり，辞書中の規則は，その単語が文中に現れたときだけに用いられる特別な文法規則であると考えることができる。このため，非常に柔軟な処理が可能になる。

GRADEでは，後者のことも可能になっている。即ち品詞や意味マーカ等の情報は辞書中にパラメータとして表現し，文法規則本体では扱いにくい，個々の単語に依存する処理，たとえば，格フレームのマッチング[3)]などを辞書中の規則（文法規則本体と同一の形式）で表している。この場合，辞書規則をどの時点で起動するかが問題である。GRADEでは，本体の文法規則から，辞書中の規則を起動することにより処理を行ってい

```
IADVP.RR ;                          " PICK UP A UNIT 'IADVPS' "

 MATCHING_INSTRUCTION ;
   L_TO_R ;
   ORDER(2,NOSKIP) ;

 MATCHING_CONDITION ;
    %( IADVPS HA1 ) ;
        IADVPS : ANY(  ~(%( HA2 ) | %( CONJ ) )  ) ;

 SUBSTRUCTURE_OPERATION ;      " PARSE A UNIT 'IADVPS' "
   HA1.LEX <- 'LLLL' ;
   @IADVPS <- CALL-DIC( IADVPS.LEX TRANSFER %( IADVPS HA1 ) LIST ) ;
   HA1 <- @IADVPS ;                  ← この部分で辞書規則を呼ぶ

 CREATION ;
 IF @IADVPS = NIL ;
          THEN RULE_FAIL ;
          ELSE %( @IADVPS )  ;              "ELSE %( ( @IADVPS ) ) ; "
          @IADVPS.LEX <= 'PPPP';
 END_IF ;

END_RR.IADVP ;
```

図10 辞書規則を起動する例

る。辞書規則を起動する部分のGRADEの記述を前頁の図10に示す。この例では，IADVPの表層上のつづり（LEXで指定）をキーとして辞書引きを行い，その中のTRANSFERという規則を，IADVPとHA1をリスト（木構造の列）にしたものに対して適用することを示している。このように，一つの単語に対して複数の辞書規則が記述できる。

7. GRADEの構成と実行環境

GRADEの構成を図11に示す。4で述べたように，GRADEで記述された文法は，まずトランスレータにより内部表現（LISPの式）に変換される。次に，この内部表現と入力文の形態素解析の出力とを入力として，実行部が文法の適用を行う。そして，処理結果の木構造が形態素合成プログラムに送られる。

GRADEは，京大大型計算機センタFACOM M-382上のUTILISPとLisp Machine Symbolics 3600上のZetalisp上で開発を行っている。トランスレータと実行部はUTILISPで実現されており，Zetalisp上では，UTILISP互換用パッケージ等によりUTILISPと同一のプログラムで実行している。

現在は，トランスレータ，実行部の改良と共に，文法開発支援プログラムの設計，開発を行っている。Symbolics 3600上での実行例を図12に示す。

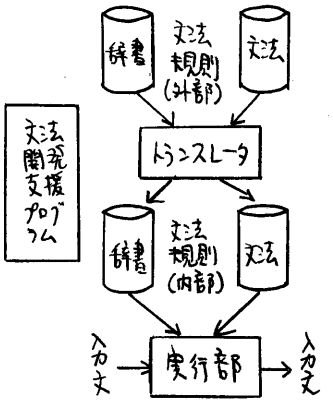
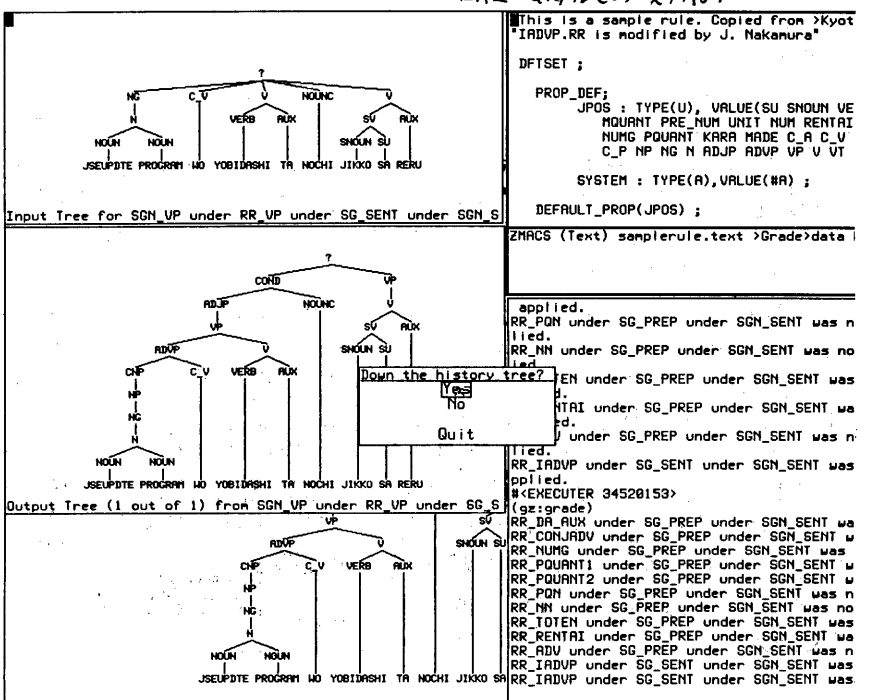


図11 GRADEの構成

8. まとめ

本報告では，機械翻訳のための文法記述用ソフトウェアGRADEについて述べた。GRADEは，部分文法の考え方を用いており，その結果，文法規則の適用の仕方を細かく制御でき，あいまいさの扱いを柔軟にできるようになっている。また，個々の文法規則を木構造の変換規則として表現しており，翻訳のすべての過程を統一的に記述できる。更に，辞書中にも文法規則が記述できるので，例外的な処理も容易にできるようになった。今後は，実際の文法をGRADEで記述し，その記述能力を確めると共に，文法を開発する上での支援プログラムの充実を行う必要がある。

図12 GRADEの実行例



参考文献

1) Boitet, C., Automatic Production of CF and CS analyzers using A General Tree Transducer, universite scientifique et médicale de Grenoble, 1979
2) GRADEマニュアル，長尾研内部資料，1983
3) 辻井潤一，日本語構文解析，情報処理学会自然言語研究会資料，1983.7
4) 長尾真，科技庁機械翻訳プロジェクトの概要，同上
5) 中村順一，Lisp Machineのpackage機能について，内部資料，1983