自然言語処理 38－6
（1983. 7. 15）

# 変換処理過程の基本設計

西田豊明
（京都大学工学部）

内容概要
本稿では，日英機械翻訳システムの変換・生成過程の基本的な処理方式について述べる．変換過程では依存格構造を中間構造として用いて，日本語の依存格構造から英語の依存格構造への変換を行なう．本方式の特徴は次のようである：①意味的に深いレベルの中間表現の使用により，言語の表層的な性質にとらわれない処理ができる．②システム作成とデバッグを容易にするため，変換過程を小さざみなステップに分けた．③訳語選択・構造変換はシステムで標準的に用意した文法以外に単語毎に定義されたサブグラマーによっても行なうことができ，柔軟性のある処理ができる．
変換・生成処理は文法記述用言語GRADEを使って記述する．「なる」,「行なう」などの特殊な語の変換規則はGRADEを使って直接記述するが，そうでない語については固定フォーマットで変換規則を与える．

## 1. はじめに

日英機械翻訳システム[1]において，変換部では日英両言語間の差異を埋めるために多くの処理を行なう必要がある．変換部を設計するとき，日本語の解析結果をもれのないように英語に変換することがまず重要である．比較的短い期間でシステムを実現することも考え，次のような設計目標をたてた．

(a) 単純さ，デバッグのしやすさ．

(b) 意味処理：日本語と英語は大変異なった言語であるから，意味情報を使って変換しないと高品質の翻訳は得られない．

(c) 例外処理：現段階で計算機にのせることのできる変換文法はあらゆる現象をカバーできるわけではないから，例外的な，単語個別の変換法もシステムに組込むことができるようになっていなければならない．

(d) 効率的な辞書作成：分野をある程度限定すると一意的に翻訳できる語や構造変換を要しない語がかなり多くなる．このような語に対する変換辞書作成作業は，例外処理や複雑な構造変換を要する語と区別して効率的に行ないたい．

このような目標を達成するため以下のような設計方針をとった．

(a)… 変換・生成部を小さざみなステップに分ける．
(b)… 依存格構造を中間表現に用いる．
(c)… 単語毎の変換規則を組込めるようにする．
(d)… 固定的な訳し方をする語に対して固定フォーマットを作る．

変換生成処理手順の概要を図1に示す．依存格構造レベルで，日本語→日本語，英語→英語の「ぐるぐる回り」の部分を設けて，heuristicな規則を適用して日英変換部の負担の軽減と質の向上を図っている．変換・生成の処理手続は本プロジェクトで開発した文法記述用言語GRADE[2]で記述する．

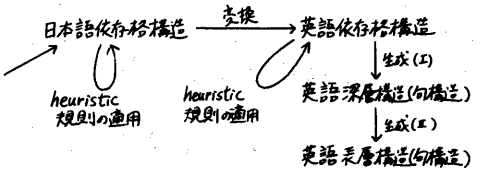


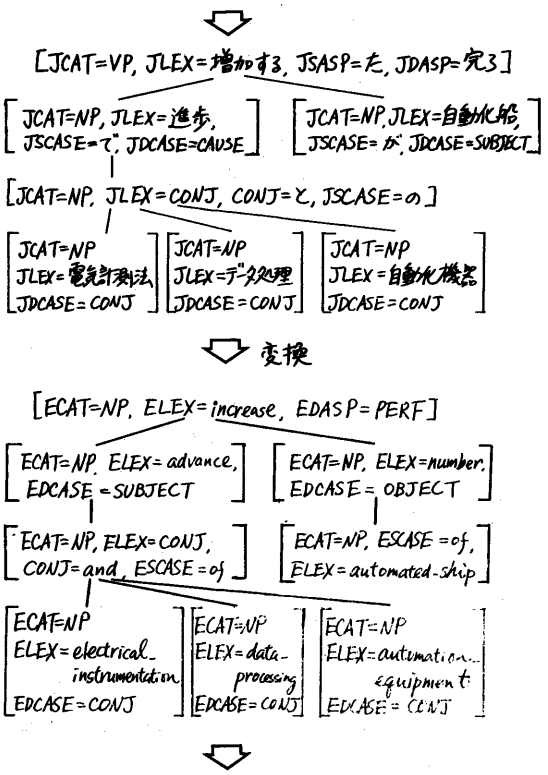
図1. 変換・生成処理の概念図．

†) 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」の一部として行なったものである．

## 2. 依存格構造の使用

GRADEでは依存格構造はプロパティをもつノードからなる木構造として実現する。依存格構造は日本語，英語についてそれぞれ定義し[3]，変換を行なう。

(例) 電気計測法，データ処理，自動化機器の進歩で自動化船が増加した。

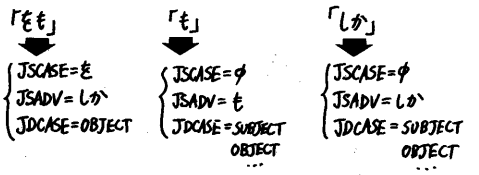



Advance of electrical instrumentation, data processing, and automation equipment has increased the number of automated ship.

この例に示したように依存格構造では日本語－英語の統語上のちがいが吸収され，明示的，暗黙的な述語－引数関係が格構造で表現されるので，訳語選択と意味的な構造変換が見通しよく行なえる。

日本語，英語の依存格構造の各ノードに与えるプロパティには，文法カテゴリ，格ラベル，テンス，アスペクト，モーダルなどに関するものを用いる。各々表層レベルと深層レベルを設け，解析で深層格が決まらなければ表層格情報に基づいて変換を行なう。

(例)




## 3. 変換処理手順

変換処理は，多少効率を犠牲にしても単純で見通しがよくなるように設計している。変換文法では，与えられた依存構造木を何回もスキャンしながら少しずつ変換してゆく。変換は構造のかわらないところから順に処理してゆく。従ってbottom upの処理が行なわれる。

変換手順の概要を図2に示す。処理の各ステップはGRADEのサブグラマーに対応づける。

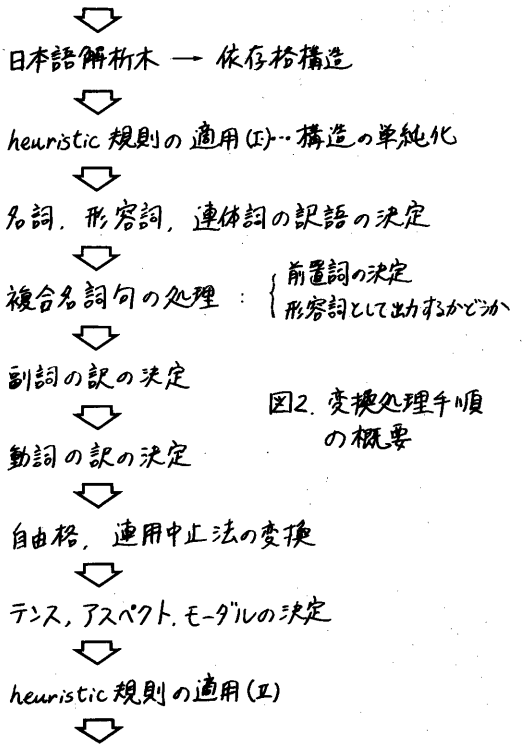
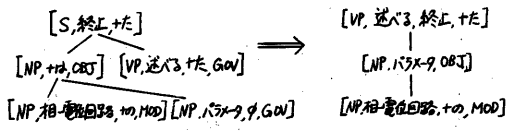



図2. 変換処理手順の概要

### 変換の準備

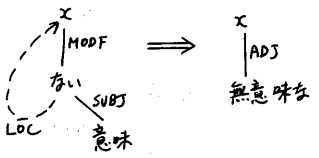
日本語解析木を依存格構造に変換する。




heuristic 規則の適用 (I)

日本語の複合構造で，単純化できるものは一つにまとめる．

(例)

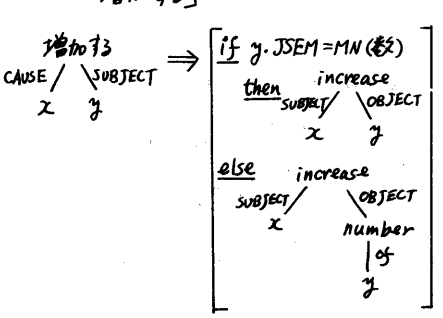



訳語選択と構造変換

訳語は格構造と各種のプロパティを参照して決定する．日英変換部では標準的な処理としてまず訳語選択を中心とした処理を行ない，少なくとも何らかの英語表現が生成されることを保障する．

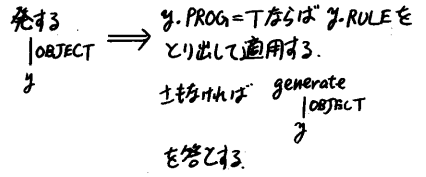
「なる」や「する」など，非常に広い使われ方がある語については単語個別の変換規則を与えておいて，変換時にチェックし，呼び出す．このような場合，動詞の辞書項目中に，その動詞がとりうる名詞句のあらゆる場合について条件わけを行ない，変換法を記述するというのは大変である．ここでは逆に，名詞句の辞書項目の方に，その名詞句が引数として参照される場合の動詞パターンを書いておいて呼出すようにしている．

(例1)　「増加する」



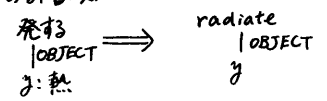

(例2)　「発する」

・「発する」の辞書項目：

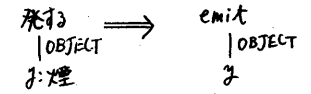



・「光」の辞書項目：


・「熱」の辞書項目：




・「煙」の辞書項目：




heuristic規則の適用 (II)

生成された英語の依存格構造を調べ，単純化できるものがあれば一つにまとめる．

(例) do harm → harm

文法設計の自由度

実際の日英変換においては，与えられた現象をどの段階で取扱ってよいか決めにくいものが多い．例えば，「害を与える」，「影響を及ぼす」のような表現は，日本語の「イディオム」ともとれるし，英語に変換するときの問題ととらえることもできるし，また，ここで示したように，"harm" vs "do harm" という英語内部での問題として取扱うことも可能である．

変換過程では，与えられた訳語選択や構造変換を実現するためのステップは唯一ではなく，問題の性質に応じて変えることができる．従って文法の設計者は「害を与える」の変換を別のステップで行なってもよい．日本語内での変換，日英の変換，英語内での変換は重

複していてもよい。

一般には，heuristic規則は補助的なもので，それがなくても許容できる英語が生成されるが，それがあるともっとよい出力が得られるような文法が望ましい。

## 4. 日英変換のための辞書記述形式

日英変換のための辞書項目のうちのほとんどは，科学技術文献の範囲では複雑な構造変換を要さず，比較的一意的に訳語を割当てることができると考えられる。このような語については，固定されたフォーマット形式を用いて変換規則を記述する†

固定フォーマットの日本語側は，与えられた単語のそれぞれの格パターンに対応する。さらに，各格スロットに入っている値に応じて複数個の英語の格構造を対応づけることができる。固定フォーマットはこのように一段までの格構造変換は記述できるので，あまり大きな構造変換を要しない単語の変換はこのフォーマットでかなり記述できる。

この固定フォーマット形式で記述された辞書項目はコンパイラによってGRADEの規則に落とし，他のGRADE規則と同様に使用する。

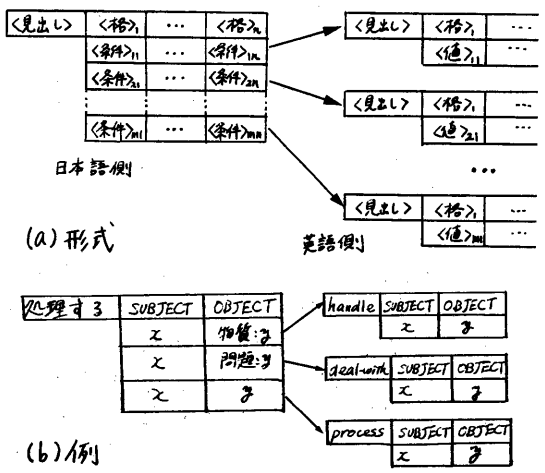



図3. 固定フォーマット

† この部分は(株)日本コンベンションサービスの高井貞治が担当している。

## GRADEで直接書く

「行なう」「なる」など，比較的大きな構造変換を要する語については文法記述用言語GRADEで直接辞書記述を行なう。これにより，かなり複雑な構造変換を行なう辞書規則が記述できる。

## 5. 英語の生成処理

英語の生成処理の過程では，まず英語の依存構造から深層の句構造を生成し，次に受動化や並列化などの変形操作を行なって表層の句構造を生成する。生成処理も変換処理と同様にGRADEの逐次的な適用によって行なう。与えられた依存構造を上から下に(top downに)変換してゆく。図4に処理の流れを示す。

## 6. GRADEによる変換・生成手順の記述

以上述べた変形・生成処理手続きをGRADE上のどのようなコードで実現するかについて，いくつか例を示す。

### 動詞の変換

動詞をキーにして辞書中のサブグラマーを呼出す。

動詞の変換を行なうサブグラマー

```
T_JE_VP.SG ;
    DIRECTORY_ENTRY ;
      OWNER(TOYO-AKI.NISHIDA) ;
      VERSION(V01) ;
      LAST_UPDATE(03/07/83) ;

    SG_MODE ;
      ORDER(2) ;
      DETERMINISTIC ;

     RR_IN_SG ;
        T_JE_VP_NP ;
       T_JE_VP ;

END_SG.T_JE_VP ;
```

第1番目の書換え規則 (RR)

```
T_JE_VP_NP.RR ;
      MATCHING_INSTRUCTION ;
         ORDER(2,SKIP) ;
         TREE ;
      MATCHING_CONDITION ;
        %( (VP1 #1 NP1 #2) );
           NP1.JPROG = 'T';
      SUBSTRUCTURE_OPERATION ;
        @X  <= CALL-DIC(NP1.JRULE TRANSFER %( VP1 ));
      CREATION ;
        %( @X  ) ;
      END_RR.T_JE_VP_NP ;
```

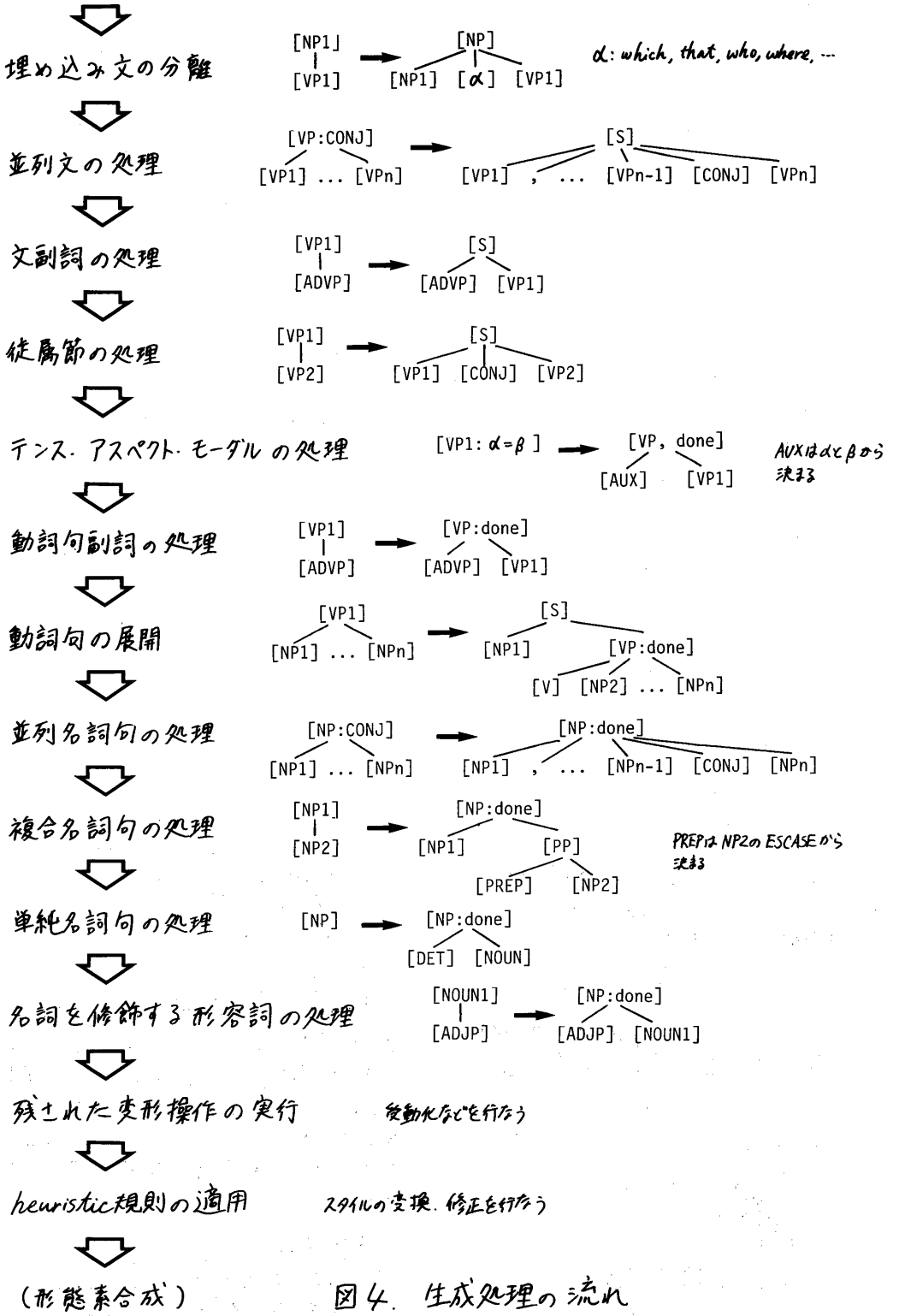

埋め込み文の分離
[NP1]
[VP1]
[NP]
[NP1] [α] [VP1]
α: which, that, who, where, ...
並列文の処理
[VP:CONJ]
[VP1] ... [VPn]
[S]
[VP1] , ... [VPn-1] [CONJ] [VPn]
文副詞の処理
[VP1]
[ADVP]
[S]
[ADVP] [VP1]
従属節の処理
[VP1]
[VP2]
[S]
[VP1] [CONJ] [VP2]
テンス・アスペクト・モーダルの処理
[VP1: α=β ]
[VP, done]
[AUX] [VP1]
AUXはαとβから決まる
動詞句副詞の処理
[VP1]
[ADVP]
[VP:done]
[ADVP] [VP1]
動詞句の展開
[VP1]
[NP1] ... [NPn]
[S]
[NP1]
[VP:done]
[V] [NP2] ... [NPn]
並列名詞句の処理
[NP:CONJ]
[NP1] ... [NPn]
[NP:done]
[NP1] , ... [NPn-1] [CONJ] [NPn]
複合名詞句の処理
[NP1]
[NP2]
[NP:done]
[NP1]
[PP]
[PREP] [NP2]
PREPはNP2のESCASEから決まる
単純名詞句の処理
[NP]
[NP:done]
[DET] [NOUN]
名詞を修飾する形容詞の処理
[NOUN1]
[ADJP]
[NP:done]
[ADJP] [NOUN1]
残された変形操作の実行
受動化などを行なう
heuristic規則の適用
スタイルの変換・修正を行なう
(形態素合成)

図4. 生成処理の流れ

第2番目の書換え規則(RR)

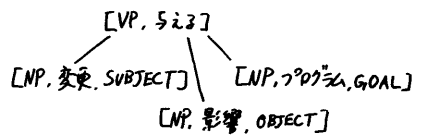
```
T_JE_VP.RR ;
    MATCHING_INSTRUCTION ;
      ORDER(2,SKIP) ;
      TREE ;

    VAR_INIT ;
      @X ;
    MATCHING_CONDITION ;

      %( VP1 );
         VP1.DONE =¬ 'T' ;

    SUBSTRUCTURE_OPERATION ;
      @X <= CALL-DIC(VP1.JLEX TRANSFER %( VP1 )) ;

    CREATION ;
      %( @X ) ;
        @X.ECAT <= 'VP' ;

END_RR.T_JE_VP ;
```

第1番目の書換え規則では、動詞の格スロットを埋める名詞句中に特殊な変換を行なう規則が入っているかどうかを調べている。もしあればそれを優先的に適用する。例えば「Xに影響を与える」を"affect X"と訳そうとするとき、「与える」の目的格に入る値によって細かく場合分けするのではなく、「影響」の辞書項目中に、それが「与える」の目的格として使われたときは動詞は affect,「受ける」の目的格として使われたときは…、というような規則を与えるようにしている。

これが適用されなかったときは第2番目の書換え規則が呼出され、格情報によって訳を決める。

(例)　「変更がプログラムに影響を与える」

〈依存構造木の概略〉




〈GRADE内部のLISP表現〉

```
(((JCAT (VP)) (JLEX (ATAERU)))
 (((JCAT (NP)) (JLEX (HENKOU)) (JDCASE (SUBJECT))))
 (((JCAT (NP))
   (JLEX (EIKYOU))
   (JPROG (T))
   (JRULE (ATAERU_EIKYOU))
   (JDCASE (OBJECT))))
 (((JCAT (NP)) (JLEX (PROGRAM)) (JDCASE (GOAL)))))
```

} 辞書項目中に書換規則があることを示す

〈名詞の辞書中のサブプラマー〉

```
   "ATAERU_EIKYOU"
  ATAERU_EIKYOU_TRANSFER.SG ;
     SG_MODE ;
       ORDER(2) ;
       DETERMINISTIC ;
     RR_IN_SG ;

       ATAERU_EIKYOU_TRANSFER ;
     END_SG.ATAERU_EIKYOU_TRANSFER ;
```

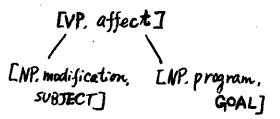
(右上につづく)

(左下から)

```
ATAERU_EIKYOU_TRANSFER.RR ;
   MATCHING_INSTRUCTION ;
     LEVEL(0,0) ;
     TREE ;
   MATCHING_CONDITION ;
     %((VP1 #1 NP1 #2 NP2 #3)) ;
        NP1.NP2 : DISORDER( LR );
     VP1.JLEX = 'ATAERU' ;
     NP1.JLEX = 'EIKYOU' ;
     NP1.JDCASE = 'OBJECT' ;
     NP2.JDCASE = 'GOAL' ;
   CREATION ;
     %((VP1 #1 NP2 #2));
     VP1.ELEX <= 'AFFECT' ;
     VP1.DONE <= 'T' ;
     NP1.EDCASE <= 'SUBJECT' ;
     NP2.EDCASE <= 'GOAL' ;
   END_RR.ATAERU_EIKYOU_TRANSFER ;
```

〈結果〉

```
(((JCAT (VP)) (JLEX (ATAERU)) (ELEX (AFFECT)) (DONE (T)))
 (((JCAT (NP))
   (JLEX (PROGRAM))
   (JDCASE (GOAL))
   (EDCASE (GOAL))
   (ELEX (PROGRAM))
   (ECAT (NP))))
 (((JCAT (NP))
   (JLEX (HENKOU))
   (JDCASE (SUBJECT))
   (EDCASE (SUBJECT))
   (ELEX (MODIFICATION))
   (ECAT (NP)))))
```




並列名詞句の生成を行なう書換規則

```
G_E_NP_C_NP.RR ;

    MATCHING_INSTRUCTION ;

      L_TO_R ;
      ORDER(2,SKIP) ;
      TREE ;

    VAR_INIT ;
      @PRE_NPS ;
    MATCHING_CONDITION ;
      %( (NP1 NPS) );
        NP1.ELEX = 'CONJ' ;
        NPS : %( PRE_NPS NP21 NP22) ;
        PRE_NPS : ANY( %( NP )) ;

      SUBSTRUCTURE_OPERATION ;
         @PRE_NPS <= CALL-SG(G_E_NPS %( PRE_NPS )) ;
    CREATION ;
      %( (NP3 @PRE_NPS NP21 CONJ1 NP22) );
      NP3.ECAT <= 'NP' ;
      NP3.EDCASE <= NP1.EDCASE ;
      NP3.ESCASE <= NP1.ESCASE ;
      NP3.CMXNP <= 'T' ;
      CONJ1.ELEX <= 'AND' ;
      CONJ1.ECAT <= 'CONJ' ;
```

(図4の8番目のステップ)
ここでサブプラマー G_E_NPSは
NPのあとにコンマを入れる:
([NP])→([NP] [CONJ]).
,

## 7. まとめ

本稿では日英機械翻訳システムの変換生成処理の基本設計について述べた。本稿で述べたシステムは一部実際にGRADEによって記述され稼動している。現在変換文法の詳細について検討するとともに、変換辞書作成作業を行なっている。

## 参考文献


[1] 長尾：科技庁機械翻訳プロジェクトの概要
[2] 中村：文法記述用ソフトウェアGRADE
[3] 坂本：格構造を中心とした用言と付属語辞書


} 本研究会資料