自 然 言 語 処 理 83－6
（1991. 5. 16）

論理文法に基づく文解析文生成システム

林達也* 坂巻利哉** 神矢浩治**
* 富士通研究所
** 富士通ソーシアルサイエンスラボラトリ

筆者等はこれ迄，統語的側面に焦点を当てて論理文法に基づく自然言語処理の枠組みについて検討して来た．本稿ではこの枠組みをベースにして，自然言語の統語及び意味を融合した文解析，文生成の枠組みについて考察する．コンピュータ上への実現は，統語処理の場合と同様に論理型言語をベースにしている．これは実世界イメージに近い形で対象システムを構築出来る事と同時に，統語処理と意味処理との整合性が良いからである．本システムの狙いは次の通りである．

(1) 宣言的記述と高次処理機構
(2) 曖昧性の許容と早期絞り込み
(3) 中間表現の大域的把握
(4) 情報のオブジェクトアプローチによるモデル化

Sentence analysis and generation system based on logic grammars

Tatsuya Hayashi* Toshiya Sakamaki** Kohji Kamiya**
* Fujitsu Laboratories LTD.
1015 Kamikodanaka Nakaharaku Kawasaki, 211, JAPAN
** Fujitsu Social Science Laboratory LTD.

The authors have previously investigated frameworks for natural language processing based on logic grammars, placing emphasis on syntactic aspects. This paper describes a framework for sentence analysis and generation. Our framework integrates the syntax and semantics of natural language, and is based on the previously developed systems. As in the case of syntactic processing, the implementation of the integrated system is based on a logic programming language. This implementation method allows us to build systems in an environment similiar to the real world, and in additon, is suitable for merging syntactic processing and semantic processing.
The objectives of this system are as follows:-

(1) declarative expression and high-order processing mechanism,
(2) allowance for ambiguity and its early resolution,
(3) global understanding of interlingua representation,
(4) modelling of required information by object approach.

1．まえがき

筆者等はこれ迄，統語的側面に焦点を当てて論理文法に基づく自然言語処理の枠組みについて検討して来た．本稿ではこの枠組みをベースにして，自然言語の統語及び意味を融合した文解析，文生成の枠組みについて考察する．

コンピュータ上への実現は，統語処理の場合と同様に論理型言語をベースにしている．これは実世界イメージに近い形で対象システムを構築出来る事と同時に，統語処理と意味処理との整合性が良いからである．

本システムの狙いは次の通りである．

(1) 宣言的記述と高次処理機構
(2) 曖昧性の許容と早期絞り込み
(3) 中間表現の大域的把握
(4) 情報のオブジェクトアプローチによるモデル化

以下では機械翻訳に重点を置き，2.で文解析について考察し，3.で文生成について述べる事にする．

2．文解析

意味処理は前に述べた限定文脈依存文法（YAPXR）或いは句構造文法(YAPXRⅡ)[1~7]をベースとし，概念モデル及び文意構造を導入して行う．概念モデルは対象世界に登場する概念の集合と概念間の上位下位，部分全体，成分ならびに深層格関係を体系的に集めたものである．

各々の概念はクラスオブジェクトと見做され，通常のオブジェクト指向言語[8,9]に見られる上位下位関係の他に，概念間の部分全体，成分ならびに深層格関係等がオブジェクト間の関係として新たに追加されている．

文意構造は原文の意味内容を概念モデルの投影によって抽出したもので，原文中の単語概念（概念モデルのインスタンス）とそれらの深層的関係から成る．文法カテゴリには意味的属性を与える事が出来る．そして，我々の横型トップダウン解析手法の特徴により，文法規則右辺の任意の場所に意味的制約を与える事が出来るようになっている．これにより曖昧性の許容と同時に，その早期絞り込みを行う事が出来る．

意味的制約は概念モデルや文意構造を参照する述語を用いて規定する．

概念モデルはATLAS[10~12]の世界モデルを，将来，高度な意味処理や文脈処理が可能なように，オブジェクトアプローチの視点で捉え直したものである．又，原文の意味内容を表現する文意構造の図式的イメージとしては，ATLAS の概念構造を拡張して得られる構造化された意味ネットワークで表現する．

2．1　意味的制約

意味的な制約はあらかじめ概念モデルを構築しておき，それを参照する述語を適宜文法規則に付加することで規定する．述語によって入力文から抽出された意味内容は文意構造（構造化意味ネットワーク）で表現される．文意構造の構築は，述語及び引数によって徐々に行なわれる．例えば，

```
vp(WDV, MPV) →
      [vt]  (WDV, MPV) np(WDN, MPN)
        {sem(MPV, MPN, obj, p),
         putcs(WDV,WDN,obj,p)} .          (1)
```

を考えよう．ここでWDV(WDN)はvt(np)が保持している単語概念を表わし，MPV(MPN)はその概念を包含する最小の分類概念で意味素性と呼ぶ．いずれも，辞書情報の一部である．又obj は，WDV とWDN ないしMPV とMPN の深層的関係の種類を示す．

関係は向きを有しており，pは左から右，nは右から左を表わす．上の例では，sem述語により概念モデルを参照して，MPV とMPN がobj の関係で結合されているならば，putcs 述語により文意構造に(WDV, WDN, obj, p) を付加すべきことを規定している．概念モデルや文意構造については後で述べるが，ここで簡単に言えば，前者は意味素性又はその上位概念間の深層的関係を示すオブジェクト体系で，後者は単語概念間の深層的関係を示すオブジェクト構造である．

例えば，入力文「(He) sees the swan」は上の規則に合致し，WDV, MPV, WDN, MPNはそれぞれsee, pwork（身体活動），swan, bird（鳥類）である．そして，概念モデルを参照して(pwork, bird, obj, p) が成立すれば，文意構造に (see, swan, obj, p)が付加される．

更に，WDV(see)，MPV(pwork)をvpが保持すべき単語概念，意味素性として規定している訳である．関係や向きはあらかじめ指定せず変数のままでも良い．又関係を一意にではなく，ある範囲に含まれていれば良いということで，集合として指定することが出来る．その場合は，

sem(MP1, MP2, R(r1, r2, r3), p)

のように記述する．この時は，概念モデルを参照して得られた関係Rがr1，r2，r3のいずれかであれば成功し，(WD1, WD2, R, p)(但しR=r1 or r2 or r3)を文意構造に putcsを用いて付加する．

次に例えば，「子供の手を引く」，「賭事から手を引く」などのように慣用表現が関係すると，2 項関係だけでは「手を引く」の意味が把握できない場合がある．このような時は，sem 述語を必要なだけ使用し，曖昧性を解決すればよい．

2．2　トレース（痕跡）処理

外置現象を正しく処理する為には，トレース（痕跡）に対して適切な統語意味属性が伝えられるようにしておく必要がある．例えば，

「The cat that likes fishes」の　場合，「The cat」の属性情報がトレースに伝えられなければ，関係節の意味を正しく抽出することができない．本システムでは次のように取り扱う．

```
np  → [det ]  [n ]  ( WDN, MPN)
      rel(/WDN,MPN/).                    (2)

rel(/WD,MP/)  → [relpn ]
      s/( ～np(WD,MP)).                  (3)
```

ここで，規則(3) のs，np はそれぞれスコープ，スラッシュと呼ばれ，s の構成要素npがトレースに対応する事を示す．

上記のようにスラッシュnpに引数を与えることにより，二つの規則に現われるrel を仲介として，規則(2)の［n］の属性情報 WDN，MPN がスラッシュnpに伝えられる．

マルチトレース（同一の先行詞に複数個のトレースが対応する）の場合は，cs//s/(～np(・・・))のように記述する．ここでcsはドメインと呼ばれ，この表現はcsの構成要素s が出現する度に，s の構成要素npがトレースに対応付けられる事を示す[2]．

2．3　文意構造

機械翻訳を前提にした場合通常，入力単語が担っている概念とそれらの概念間の関係の有無ならびに関係の種類を把握することを以て，入力文の意味内容を抽出したと考える立場が採られている．本論文においても，概念の基本単位は単語レベルの塊りであり，関係は深層格レベルで設定している．意味内容を表現する文意構造は，特定の言語に依存しない多言語翻訳向きの形式になっている．尚，入力文の構造を捉え，文意構造の変形や訳文生成の処理を容易に行なえるように，文意構造の構造化を計っている．又，訳文生成に当って入力文の語順を考慮できるように，単語概念の属性として語順情報を付加するものとする．

例を示そう．

「The mouse that the cat chases squeaks」に対する文意構造の図式的イメージは図1のようになる．

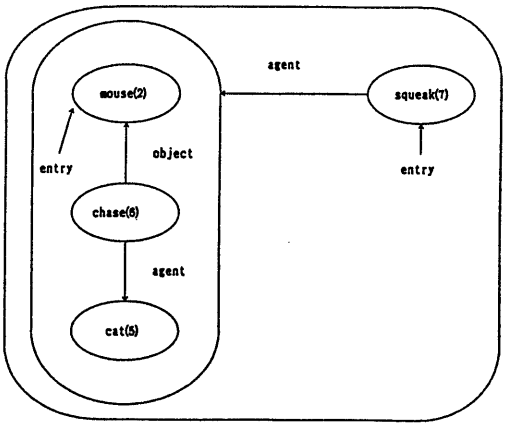

図1　文意構造の例

図1に於いて，( ) 内が語順情報，閉曲線が構造（カプセルと呼ぶ）を表わす．entry は，カプセルに於ける中心概念を表わす．つまり，文意構造は図式的には構造化された意味ネットワークである．カプセルは複合オブジェクトということになる．

2．4　概念モデル

入力単語概念間の関係を把握する為に，概念モデルが必要となる．

単語概念そのものの許される組合せは量的爆発を生じるので，最低限，意味素性間の許される深層的関係の集合としてモデルを構築する必要がある．しかしこれでも，一般にはモデルが巨大化する可能性がある．そこでこれを避ける為，上位下位関係，同義類義関係[13,14]や更に部分全体関係，成分関係，因果関係をモデルに導入している．

実現された概念モデルは，（MC1，MC2，R，D）なるタップルの集合となる．ここで，MC1，MC2は概念を表わす．Rは深層格関係に加えて，super，part of，material，effectとsynonym，similar を含む．super(part of，material，effect) は，MC1 とMC2 とがDの向きに上位関係（部分関係，成分関係，因果関係）にある事を示す．synonym，similar はMC1 とMC2 とが同義類義関係にあることを示す．又Dは，従来の関係の　向き（p，n）に加えて，否定を表わすｆも含むものとする．

2．5　実現

上述した統語意味表現形式を論理型言語に変換することについて考える．

基本的には，YAPXR 或いはYAPXRⅡと同様に規則右辺の構文要素に対応するホーン節の集合から構成される．但し，「解析」，「省略」，「引数」の各スタックに加えて，入力文の意味内容をタップルの集合として格納する為に，文意構造スタック（CS）を必要とする．

(1)　引数の受け渡し

述語間の引数の受け渡しは統語処理の場合と同様に，引数スタックを用いて行なわれる．

文法規則に於いて，構文要素の引数が後続の要素又は意味述語の引数中に現われていれば，当該ホーン節でその引数を引数スタックに残しておけばよい．規則の左辺は要素の順序を考える場合，右辺の最後に位置するものとして扱う．

(2)　意味的制約の処理

意味的制約は，規則の中で直前に位置する構文要素に対応するホーン節に含められる．

制約が規則の右辺先頭に存在してもよいが，この時は，直後の構文要素に対応するホーン節に含めればよいであろう．

例えば，

```
s(・・・) → np(・・・)
                vp(・・・) {sem(・・・)}.
```

に於いては，述語sem は，

```
vp(・・・):- !, sem(・・・), s(・・・).
```

のようにvpのホーン節に含められる．

(3)　トレース属性の処理

2.2 で示したように，規則に

```
s/(～np(WD,MP))
```

の如く引数付きのスラッシュnpが含まれていれば，省略スタックのスラッシュ型要素（g,np）を変更して（g，np(WD,MP)）のように引数付きで格納すればよい．

マルチトレースの場合は，ドメインをcsとして，スコープ・スラッシュとその引数を上記の通りとすれば，ドメイン型要素及びスラッシュ型要素を変更して，(g，np(WD,MP)),(d，cs(WD,MP)) のように引数付きで格納する．そして，トレース処理を行う度に，それら省略スタックに格納した引数を利用すればよい．

(4)　文意構造の生成

入力文に対する統語意味処理の結果，文意構造がスタックに形成されていく．

これを実現する為には，例えば2.1 で示した規則のnpの場合，次のようにホーン節を作成すればよい．

```
np ( [WDN,MPN,WDV,MPV | Li] , Lo, CSi, CSo)
 :-  sem(MPV, MPN, obj, p),
     putcs( [MPV, MPN, obj, p] , CSi, CSi2),
     vp ( [WDV,MPV | Li] , Lo, CSi2, CSo).
```

これにより，vtとnpに対応する単語概念WDV とWDNとの結合関係を示すタップルが文意構造スタックへ格納される．

(5)　意味述語

標準的な意味述語の種類と機能は次の通りである．

・mark：文意構造スタックにタップル（mark）を格納する．このタップルはスタックのこれより上に格納されたタップル（複数）をカプセル化する際の境界を示す．

・entry(WD) ：文意構造スタックにタップル (WD, nil, entry, n) を格納する．このタップルはWDが現カプセルの中心概念であることを示す．

・capsule(条件式：entry(WD):変数) ： 条件式が満たされた時，WDを中心概念として，文意構造スタックの先頭から境界までのタップル（複数）をカプセル化し，変数にユニファイする．

・putcs(WD1, WD2, R, D) ：文意構造スタックにタップル (WD1, WD2, R, D) を格納する．

・getrel(MP1, MP2, R, RV, D)：関係を一意に決めず集合Rとして与え，概念モデルを参照して得られた関係をRVにユニファイする．

・sem(MP1, MP2, R, D) ：概念モデルを参照してその真偽を返す．

・addattr(WD1, 属性, WD2)：WD1 に属性を与え，WD2 にユニファイする．

・putdet(WD1, WD2, WD3) ： WD1に属性として [det, WD2] を与え，WD3 にユニファイする．

例文に基づいて本方式の動作を見てみよう．

例文

「The books were given him by his uncle 」

解析に必要な文法を付録１に示す．

文法に記述してある通り，解析は文脈依存規則，外部引数の適用，構文・意味制約[4] を用いて，効率良く解析パスの絞り込みを行なっている．

前述の意味述語を制約式中に導入することにより，構文解析のみでは絞り切れなかった不要な解析結果の削除を可能にしている．例えば，文法(1)に於ける概念モデルの参照がそれである．

例文の文意構造は，以下に示した順番でＣＳスタックに形成されていく．（)内は使用した文法規則である．

[give, he, goal, p] ———— (1)
[give, book, obj, p] ———— (1)
[uncle, he, possessor, p] —— (6)
[give, uncle, agent, p] —— (4)
[give, nil, entry, n] ——— (2)

解析によって得られた文意構造を図２に示す．

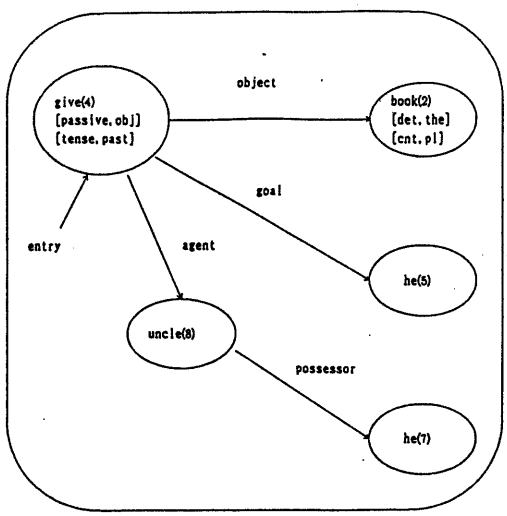

図２　例文の文意構造

３．文生成

一般に，文法モデルを解析及び生成で共通化することも一応考えられる．しかしそれは，

(1)生成に於いては原則として，概念モデルの参照をもはや必要としない．

(2)生成する文体は一定の書式に統一することが出来るので，表現形態の変動を考慮する必要はない．

(3)解析に於いては，最終的に解決されるとは言え，曖昧性を許さなければならない．しかし，生成ではその必要がない．

(4)解析に於いては任意の文を取り扱う必要があるが，生成では文意構造に対応する特定の文を対象とすれば良い．

等から，工学的見地からは問題がある．そこで本論文では，文生成の枠組みを解析用とは別個に設けることにする．

３．１　構造変換

文意構造の枠組自体は，特定の言語に依存しない多言語向きの機構である．すなわち，構造化意味ネットワークであり，ノードは単語や句（カプセルと呼ぶ）の概念を表わし，アークは概念間の関係を表わしている．しかし一般に，具体的な概念のセットと概念同志の結び付け方は，言語によって異なってくる．そこで，原言語の構造と目標言語の構造の間で変換が必要となる．

更に，文意構造変換は原言語内に於いても有効な場合がある．例えば，「賭事から手を引く」というような慣用表現に対して，原文解析で本来の文意構造を直接作成しても勿論差し支えないが，一旦文字通りの構造に留めておく事も考えられる．原文が種々の修飾句により複雑な構造になっている様な場合には，構造変換を利用して最終的な文意構造を求めた方が分り易い．

文意構造変換は，更に文脈や知識情報を反映させる場合にも必要であるが，此処ではこれ以上は言及しない．

3．2　文生成文法

文意構造から文を生成する文法形式として，一種の文脈自由文法形式の論理文法を定める事にする．その際，文生成はモジュラリティを高め見通しを良くする為に，構文構造生成，形態素生成の2段階に分けて行う事にしている．

構文構造生成規則の基本形式は次の通りである．

左辺　→　右辺．

左辺　：　［部分パターン］
　　構文要素(Y,Mode,Env)

右辺　：　右辺項・・・

右辺項：　［()　［部分パターン］
　　構文要素(Y,Mode,Env)
　　{;［部分パターン］
　　構文要素(Y,Mode,Env)} * [)]

生成規則は構文要素に引数を付与し，かつ条件部（部分パターン）を導入した形式で，条件部には｛　｝を用いて文意構造とマッチングをとるべきパターンを記述する．左辺の条件部は規則自身の適用の可否を指定する．右辺の条件部はその直後にある構文要素に基づく生成の可否を指定する．尚，条件部を直前に持たない構文要素は無条件で有効であることを示す．

条件部に於いては，文意構造全体を見通して，必要な部分構造の存在や不在を容易に指定出来るように考慮されている事も本文法の特徴である．

構文要素の第1引数Ｙは文意構造中の単純ノード又はカプセルを表わし，第2引数Ｍｏｄｅは生成モード（活用形，表層格に相当）を示す．生成モードは，end(終止)，cond(条件)，cn(連体)，cv(連用)，subj(主格)，obj(目的格)，loc(場所)，by(動作主)，manner(方法)などである．第3引数Ｅｎｖは適切な訳語選択に必要な情報を示す．

文法例を付録2に示す．尚，規則中の‘@’は，カプセル或いは修飾ノードを持つノードを表し，‘#’は単純ノードを表わす．又，終端記号は［　］で表示され，単語辞書を参照して指定されたモードで訳語を生成する．

図2の文意構造例に付録2の文法を適用すると，図3のような生成木が得られる．

入力した文意構造例は，エントリが存在するため付録2の文法規則(1)の変数Ｓとユニファイし，文法規則ｖｐを呼び出す．ｖｐの文法規則には(2)と(3)があるが，第1引数がカプセルであるため文法規則(2)は採用されない．次に文法規則(3)の適用を試みると，左辺条件部とのマッチングに成功する．右辺条件部により，変数Ｎ1，Ｎ2にはそれぞれｂｏｏｋ，ｈｅがユニファイされ，モードｓｕｂｊ，ｇｏａｌで文法規則ｎｐを呼び出す．次に，変数Ｎ3にｈｅ，Ｎ4にｕｎｃｌｅがユニファイされ，3種類の条件部から①がマッチングに成功し，モードｂｙで文法規則ｎｐを呼び出す．最後に無条件で文法規則ｖｐを呼び出す．呼び出された文法規則ｎｐ，及び文法規則ｖｐは，第1引数が単純ノードであることを確認して生成木に終端ノードを出力する．

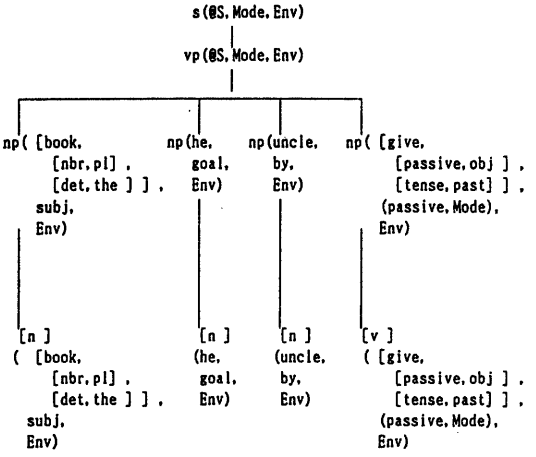

図3　生成木例

4. あとがき

本稿では，論理文法に基づき統語意味情報を融合した，自然言語処理に於ける文解析，文生成の枠組みについて考察した．

概念モデルや文意構造はオブジェクトアプローチの立場で捉えられる．即ち，各概念はクラスオブジェクトとして捉えられ，入力文の各単語が担っている意味は対応する概念のインスタンスとして扱われる．通常の計算モデルとしてのオブジェクト指向と比較した場合，オブジェクト間の上位下位関係の他に部分全体関係や成分関係，深層的関係，同義類義関係，因果関係を含むので，オブジェクト間の関係の種類が多くなっているのが特長である．

文意構造は特定の言語に依存しない多言語向きの形式を有しているが，具体的な概念のセットや概念間の関係は一般に言語によって異なってくる．従って，文意構造の変換が機械翻訳のような場合には必要となる．そこで，本稿では触れなかったが，双方向性の非手順的構造変換の枠組みについても検討する必要がある．

最後に，文意構造からの文生成について考察した．そして，工学的見地から効率性を考慮して，文解析とは別の枠組みを設定した．本論文の概念モデルや文意構造に現れるオブジェクトは，一文単位の翻訳処理の現状を反映して，情報量が少なくて済み現段階では充分に機能していない．しかし，言語外知識や文脈情報を組織化して且つ差分的にオブジェクトに蓄積できるので，例えば文章単位の次世代翻訳処理等に対する成長発展の可能性を有していると考えている．

此処で述べた幾つかの枠組みの評価が今後の課題である．

参考文献

(1) 林達也：論理型言語による構文解析法YAP について，情報処理，Vol.29，No.9，pp835～842(1988)．

(2) 林達也：拡張CFG とその構文解析法YAPXについて，情報処理，Vol.29，No.5，pp480 ～487(1988)．

(3) 林達也：YAPXの効率的実現法，情報処理論文誌，vol.30,No.10,pp 1354-1356(1989)

(4) 林達也，宮俊司，坂巻利哉，吉田健一：横型トップダウン文解析システムの実現と評価，情報処理学会，自然言語処理研究会，Vol.74- 9，pp.65-72(1989)．

(5) 林達也：文解析システムYAPXR の実現と評価，情報処理学会論文誌,Vol.31,No.7 ，pp.970-978(1990)．

(6) 林達也，宮俊司，坂巻利哉，吉田健一：句構造文法に対する効率的文解析手法，情報処理学会，自然言語処理研究会,Vol.76-2，p.8(1990)．

(7) 林達也：句構造文法に対する効率的文解析手法，情報処理学会論文誌,Vol.31,No.12，pp.1718-1726(1990)．

(8) Goldberg,A.and Robinson,D.:Smalltalk 80: The Language and its Implementation, Addition Wesley(1983)．

(9) Stroustrup,B.:The C ++ Programming Language,Addition Wesley(1986)．

(10)Hayashi, T. et al: ATLAS: automatic translation system, FSTJ, Vol.21, No.3, pp317～329(1985)．

(11)Hayashi, T. et al: ATLAS: Fujitsu's machine translation system, Proc. Asia and Pacific regional Conference on Translation, p20(1986)．

(12)大特集：機械翻訳，情報処理，Vol.26，No.10，pp1139～1236(1985)．

(13)R.L. Chapman: Roget's international Thesaurus, Harper & Row Ltd.(1979)．

(14)大野晋，浜西正人：角川類語新辞典，角川書店(1984)．

付録1　例文解析用文法

```
(1) so → s(WD, MP).
(2) s(WD, MP)  → sdec(WD, MP)
                    {entry(WD) } .
(3) sdec(WD, MP)  → sdec01(WD, MP).
(4) sdec01(WDV2, MPV)  →
      subj(WDS, MPS)
      aux(WDAU, FLEX2)
      advx(WDAD, MPAD, ADINF)
      vp(FLEX, WDV, MPV)/(~ obj(WDS, MPS))
                          (/WDS, MPS/)
       {getrel(MPV, MPAD, ADINF, RV, posi)&
        putcs(WDV, WDAD, RV, posi)&
        or(FLEX2 == ee &
          addattr(WDV,
          [[tense,present]],WDV2)
         ; FLEX2 == ed &
          addattr(WDV,
          [[tense,past]],WDV2)
         ; FLEX2 == en &
          addattr(WDV,
          [[tense,pp]],WDV2)) }
      ppx(WDPX, MPPX, PINF)
       {getrel(MPV, MPPX, PINF, RV, posi)&
        putcs(WDV2, WDPX, RV, posi) } .
(5) subj(WD, MP)  → np(WD, MP).
(6) np(CAPS, MPND)  →
     allx(WDAX, MPAX) {mark}
     ddet(WDDT, MPDT)
     nomhd(WDND, MPND)
     {sem(MPND, MPAX, RV, posi)&
       putcs(WDND, WDAX, RV, posi)&
       or(sem(MPND, MPDT, RV2, posi)&
          putcs(WDND, WDDT, RV2, posi)&
          WDND2 = WDND
       ;  putdet(WDND, WDDT, WDND2))}
     ncompx(WDNCX, MPNCX, PINF)(/WDND2, MPND/)
       {getrel(MPND, MPNCX, PINF, RV, posi)&
        putcs(WDND, WDNCX, RV, posi)}
     ncomp __tx(WDNTX, MPNTX)(/WDND2, MPND/)
       {capsule(WDNTX ¥== nil: entry(WDND2)
                             : CAPS)} .
(7) ddet(WD, MP)  → [det](CATX, WD, MP).
(8) nomhd(WD, MP)  → [n](WD, MP).
(9) aux(WD, MP)  → bep(CATX, FLEX, WD, MP).
(10) bep(CATX, FLEX, WD, MP) →
                  [be](CATX, FLEX, WD, MP).
(11) {(subj aux advx) }
     vp(FLEX, WDV2, MPV)(/WD, MP/)  →
     [vt](CATX, FLEX, WDV, MPV)
       {FLEX == en & CATX == voo}
     obj1(WDO, MPO)
     obj2(WDO2, MPO2)
       {sem(MPV, MPO, goal, posi)&
        putcs(WDV, WDO, goal, posi)&
        sem(MPV, MPO2, obj, posi)&
        putcs(WDV, WDO2, obj, posi)&
        or(WD == WDO &
           addattr(WDV,[[passive, goal]],WDV2)
        ; WD == WDO2 &
           addattr(WDV,[[passive,obj]],WDV2)
        ; WDV2 = WDV)} .
(12) obj1(WD, MP)  → obj(WD, MP).
(13) obj2(WD, MP)  → obj(WD, MP).
(14) obj(WD, MP)  → np(WD, MP).
(15) np(CAPS, MPP) →
     [pron](WDP, MPP)
      ncompx(WDN, MPN, PINF)(/WDP, MPP/)
        {mark &
         getrel(MPP, MPN, PINF, RV, posi)&
         putcs(WDP, WDN, RV, posi) }
      ncomp __tx(WDNTX, MPNTX)(/WDP, MPP/)
        {sem(MPP, MPNTX, RV2, posi)&
         putcs(WDP, WDNTX, RV2, posi)&
         capsule(WDNTX ¥== nil: entry(WDP)
                              : CAPS)} .
(16) pp(WDO, MPO, PINF)  →
     [p](PINF, CATX)  {CATX == by}
      obj(WDO, MPO).
(17) ppx(WD, MP, PINF) → pp(WD, MP, PINF).
```

付録2　文生成文法例

```
(1) {entry(Y)} s(@S,Mode,Env)
        → vp(@S,Mode).
(2) vp(#Y,Mode,Env)
        → [v](#Y,Mode,Env).
(3) {entry(Y,possive=(obj))} vp(@S,Mode,Env)
        → {(N1,Y,obj)}   np(N1,subj,Env)
          {(N2,Y,goal) } np(N2,goal,Env)
         ( {(N2,N3,possessor),(N3,Y,agent)}
                          np(N3,by,Env) ……①
         ; {(N4,N3,possessor),(N3,Y,agent)}
                        (np(N4,possessor,Env)
                         np(N3,by,Env))
         ; {(N3,Y,agent)} np(N3,by,Env) )
          vp(Y,(passive,Mode),Env).
(4) np(#Y,Mode,Env)
        → [n](#Y,Mode,Env).
```