第6章　日本語文生成システム

この章では、意味ネットワークから日本語文を生成するシステムを紹介する。意味ネットワークによる意味表現と動詞辞書が与えられれば、通常の日本語文は生成可能である。また、このシステムは埋め込み文を処理することも可能である。

6.1　はじめに

文生成は、機械翻訳システムの一部として、あるいはコンサルテーションシステムの出力インターフェースの一部として用いられている。出力インターフェースとしては、現在に至るまでテーブル形式で出力文をあらかじめ用意しておく方式が中心を占めていた。例えば、適性診断テストや学力テストの診断文、コンパイラのエラーメッセージ、アプリケーションの出力プロンプトなどがそれにあたる。

しかし、AIを応用した質問応答システムや推論システムでは、そういったテーブル形式で出力文を用意しておくことは実際上不可能である。それらのシステムでは、フレームや意味ネットワークといった推論に適した知識表現をデータとして採用している。そうした知識表現により従来のデータ処理手法では処理できなかった、一様でないデータの処理が可能になった。

データ形式が一様でない以上、テーブル形式のような単純な手法で推論や出力文の処理を行なうことはできない。そこで求められるのが、任意のシステムへ組み込むことのできる、汎用的な文生成システムである。

また、文生成の手法は機械翻訳システムを構築するときには、必ず必要になってくる。機械翻訳の最初に来る形態素解析や構文解析ほどの重要度はないが、機械翻訳の最後のフェーズである文生成の部分の質が悪いと、翻訳文に対する後処理の手間が増えることになる。

この章では、意味ネットワークからの文生成の手法について詳しく考察し、実用システムに拡張することを前提とした本格的なシステムを構築する。

6.2　日本語文生成

さて、文章を生成するときに考慮しなければならない要素は以下の2つのものに分類することができる。

① 動詞、名詞、修飾語といった文を構成する要素
② 時制、話者の判断、アスペクトといった文全体にかかる属性

文を構成する要素には、各品詞類に属する具体的な語が入ってくる。これらの語は、文章を生成した時、実際に文面に出てくる。

文全体の属性には以下のようなものがある。

① 時制(過去・現在・未来)
② アスペクト(継続・完了・結果)
③ 受身、使役
④ 話者の判断(否定、推定、希望)
⑤ 話者の働きかけ(命令、疑問、禁止)

英語の場合、こういった文の属性は、用言の語尾変化と助動詞を組み合わせるだけでなく、語順を変えることによって表現する。対して日本語の場合、これらの属性は属性値に応じた付属語(助詞、助動詞)で表現され、語順を変えることはない。

このように、日本語の文生成の手続きの中で、文全体にかかる属性の処理は、動詞を中心とした述語の部分にかかっている。述語の生成さえできれば、後は名詞句や副詞、埋め込み文といった文を構成する要素を、文法にかなった順に並べるだけで、意味の通じる文を生成することが可能である。

6.3　意味ネットワークによる中間表現

6.3.1　意味ネットワークによる表現

この章で紹介する文生成システムは、意味ネットワークで表現されたデータから文章を生成する。意味ネットワークはフレームやプロダクションシステムと並ぶ代表的な知識表現手法である。意味ネットワークは当初、人間の語句の意味の記憶モデルとして開発された。それ以後は、文章をどのように理解しているかに関するモデルの表現法として、認知心理学の分野を中心にとして利用されている。

文章理解の意味ネットワークモデルには、格文法の理論が取り込まれてい

る。格文法では、動詞とその引き数の関係の記述が中心的課題となっている。引き数に割り当てられた役割を「格」といい、例えば次のようなものが考えられている。

Agent（主格）	：動作を行なった行為者
Instrument（道具格）	：動作を行なうときに用いた道具
Object（目的格）	：動作がなされた対象
Source（始発格）	：移動動作の出発点
Goal（目標格）	：移動動作の到達点

上に挙げたものはあくまでも一つの例で、格にどのようなものがあるかについての合意はなされておらず、人によって様々の格が設定されている。

意味ネットワーク表現がどういったものであるか、また格文法理論をどのように取り入れているかを、文生成の例に用いた実際の例を用いて説明しよう。**図6-1**、**図6-2**が日本語の文生成に用いる中間表現を、意味ネットワークで表現したものである。

図6-1、図6-2がそれぞれ「哲夫が裕子が卵で昨日作った料理を一人で食べている」「昨日先生に話した目付きの悪い男が先ほどきた」という文章の意味ネットワーク表現である。図中で、枠で囲ってある部分は「ノード」、ノードを結んでいる線は「リンク」と呼ばれている。意味ネットワークは、「左側のノード」の「リンクで示される属性」の値は「右側のノード」と読むことができる。例えば図6-2においては

「話す」の「object」の値は「先生」
「男」の「adj」の値は「目付きの悪い」
「人間」の「isa」の値は「生物」

のような関係が、意味ネットワーク中に表現されている。各リンクがどのような属性を示しているかの例を以下に示そう。

A　isa　B
　　AはBの1種である

A　agent　B
　　Aの主格はBである

A　object　B
　　Aの目的格はBである

A　adj　B

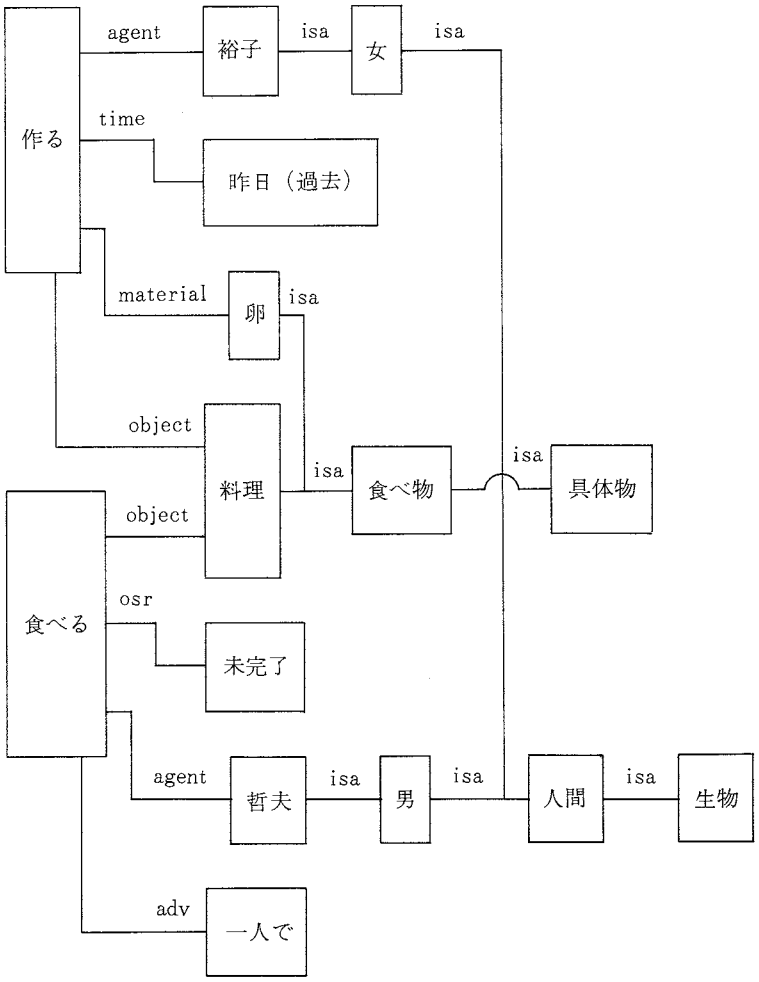

図6-1　哲夫が裕子が卵で昨日作った料理を一人で食べている

Aを修飾する形容詞はBである

A　osr　B

AのアスペクトはBである

格に関する属性以外に、修飾関係や文章では表現されていない概念間の階層関係(isa)も、意味ネットワーク中に表現されている。時制やアスペクトといった文全体にかかる意味的な属性も、同様に意味ネットワークとして表現する。図6-1、図6-2の例に出ていない主なリンクには、

locate　　場所を表わす副詞

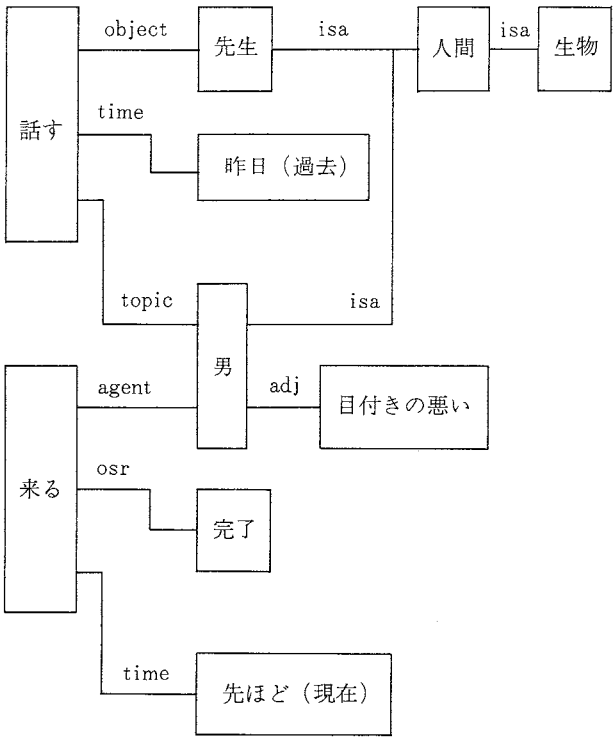

図6-2　昨日先生に話した目付きの悪い男が先ほどきた

owner　　名詞の所有者を示す

などがある。

6.3.2　動詞辞書

意味ネットワークの先頭に来ているのは、その文章の動詞である。動詞は、要素を4つもった動詞辞書を参照するために用いられる。第1要素は

[[・・・| T],T],

で、動詞の語幹部分を重リストで表わしたものがくる。・・・の部分は動詞語幹をシフトJISコードに変換したものが入る。例えば「話す」の場合は「話」のシフトJISコードである"-26526"を入れて

[[-26526 | X],X]

を語幹とする。語幹がない場合は、重リストで表現したヌルリスト

[X,X]

で表現する。

第2要素は

[行,段]

という書式で動詞の活用形が入る。例えば「話す」は、さ行5段活用なので

[さ,5]

を活用形の記述とする。

第3要素は、動詞の必要とする格に関する情報をいれる。例えば「話す」の取り得る格は、以下のような制約を持っている。

話す主体(agent)：人間
話す相手(object)：生物
話す話題(topic)：どんな要素が入ってきても構わない

この制約を辞書の中では

[[agent,が,人間,_],[object,に,生物,_],[topic,について,_,_]]

と表現する。

第4要素はアスペクト素性を表わす。アスペクト素性は現在は以下に挙げる4通りのものがある。

[+,+]　「～ている」がつかない動詞
[+,−]　「前，間，後，ながら」がつかない動詞
[−,−]　「～し終える」がつかない動詞
[−,+]　その他大半の動詞はここにはいる

例えば「話す」は「～ている」「前」「～し終える」のいずれもつけることができるので

[−,+]

となる。また、来るは「来終える」という表現を取れないので

[−,—]

となる。

図6-1、図6-2に出て来る4つの動詞の辞書を**図6-3**に示す。

```
話す([
        [[-26526 | X],X],
        [さ,5],
        [[agent,が,人間,_],[object,に,生物,_],[topic,について,_,
_]],
        [−,+]
        ]).
来る([
        [X,X],
        [か,変],
        [[agent,が,生物,_]],
        [−,−]
        ]).
作る([
        [[-29204 | X],X],
        [ら,5],
        [
                [agent,が,人間,_],
                [obj,を,具体物,_],
                [instr,によって,道具,_],
                [material,で,具体物,_]
        ],
        [−,+]
        ]).
食べる([
        [[-28600 | X],X],
        [ば,下一],
        [
                [agent,が,人間,_],
                [obj,を,食べ物,_]
        ],
        [−,+]
        ]).
```

図6-3　意味ネットワーク中に出て来る動詞の辞書

6.3.3　意味ネットワークのProlog表現

図6-1、図6-2の意味ネットワークをPrologで記述したものを、それぞれ図6-4、図6-5に示す。

```
link (time,[述語,作る],[過去,昨日]).
link (obj,[述語,作る],料理).
link (material,[述語,作る],卵).
link (agent,[述語,作る],裕子).
link (isa,裕子,女).
link (isa,女,人間).
link (osr,[述語,食べる],未完了).
link (agent,[述語,食べる],哲夫).
link (obj,[述語,食べる],料理).
link (adv,[述語,食べる],一人で).
link (isa,哲夫,男).
link (isa,男,人間).
link (isa,人間,生物).
link (isa,料理,食べ物).
link (isa,卵,食べ物).
link (isa,食べ物,具体物).
```

図6-4　図6-1の意味ネットワークのProlog表現

```
link (time,[述語,話す],[過去,昨日]).
link (topic,[述語,話す],男).
link (obj,[述語,話す],先生).
link (isa,先生,人間).
link (isa,人間,生物).
link (adj,男,[adj,目付きの悪い]).
link (time,[述語,来る],[現在,先ほど]).
link (osr,[述語,来る],完了).
link (agent,[述語,来る],男).
link (isa,男,人間).
```

図6-5　図6-2の意味ネットワークのProlog表現

述語ノードはそれが述語であることを示すために

[述語,Node]

と表記する。副詞を表わすノードは

[adv,Node]　　Node には副詞を表わすアトム
[time,Node]　　Node には副詞を表わすアトム
[locate,Node]　　Node には副詞を表わすアトム

の3種類のものがある。形容詞を表わすノードは

[adj,Node]　　Node には形容詞を表わすアトム

という書式で書くことになっている。

次に、時制などの文に関する属性についての書き方を説明しよう。「れる、られる」「せる、させる」で表現される受身、使役、可能といった属性は、「命題」というリンク名で以下のように表現する。

link (命題,[述語,Node],X).
　X には「受身」「使役」「可能」のいずれかが入る

アスペクト、時制については

link (osr,[述語,Node],X).
　X には「完了」「未完了」のいずれかが入る
link (time,[述語,Node],X).
　X には「[過去,Adv]」「[現在,Adv]」「[未来,Adv]」のいずれかが入る。Adv には時を表わす副詞が入る。

と記述する。話者の判断については

link (話者の判断,[述語,Node],X).
　X は[Mode,Time]という書式で、
　Mode には

　打ち消し・・・・(ない)
　希望・・・・・・(たい)
　推定・・・・・・(らしい)
　様態・・・・・・(そうだ)
　伝聞・・・・・・(そうだ)

不確定・・・・・(ようだ)
丁寧・・・・・・(ます)
意志・・・・・・(う・よう)
打ち消し意志・・(まい)

のいずれかのアトムが、
Time には

過去、現在、未来

のいずれかのアトムが入る。

話者の働きかけについては

link (話者の働きかけ,[述語,Node],X).

X には連言、選言、例示、程度、帰着点、類推、限定、仮定、原因、順接、逆接、並列、命令、疑問、禁止、確認、平叙のいずれかのアトムがはいる。

こうした文の属性は別に指定しなくても構わない。指定しない場合はデフォールトで、時制については「非過去」、その他については中立を表わすヌルリスト"[]"がはいる。

6.4　日本語生成システムのインプリメント

6.4.1　述語の生成

この節では動詞を中心とした述語の生成、次の節では埋め込み文の処理を考慮した日本語文全体の生成について考える。

日本語の文生成の手続きの中で最も手間のかかるのは、動詞を中心とした述語の生成である。述語の生成さえできれば、名詞句や副詞、埋め込み文の処理はそう難しくはない。特に日本語の用言の場合は、次に続く語によって様々の活用語尾を取るために、そのためのルールは膨大なものになる。ここで紹介する述語生成ルーチンは、本来のものからかなり簡略化してあるがそれでも相当量のルールを含んでいる。なお、今回インプリメントするシステムは「日本語情報処理」(電子通信学会)の第8章で紹介されているアルゴリズムを参考にしている。

述語の生成の例として、以下の文を考えてみよう。

「彼は飛行機で飛んでいったらしいので、....」

飛ん	でいった	らしい	ので
動詞	アスペクト&時制	判断	働きかけ

「トムは、ベンに殴られたようだった。」

殴	られ	た	ようだ	った
動詞	受身	アスペクト&時制	判断	判断の時制

述語の生成では、上に挙げた例に見られるように

動詞＋受身使役＋アスペクト&時制＋判断＋判断の時制＋働きかけ

というつながりが考えられる。

「動詞」とは「動詞の語幹＋語尾」のことで、語尾の活用形は後に続く語に支配されている。

「受身使役」は

「れる・られる」
「せる・させる」

という助動詞で表わされる。「せる」「れる」は5段活用やさ変活用の未然形に、「させる」「られる」は上一段、下一段、か変の未然形に接続する。

「アスペクト」は、英語でいう過去完了形で表わされる表現で、完了や継続、結果などを表わすための語である。アスペクトは次の時制と密接に結びついている。「時制」は、過去・現在・未来という時制のことである。「アスペクト&時制」の部分には

「た・だ」
「ている・でいる」
「ていた・でいた」
「つつある」
「つつあった」

という語がくる。アスペクトと時制に応じてどのような付属語を取るかは、動詞固有のアスペクト素性によって決められる。物事が進行中である事を表わすのに

酔いがさめつつある

という表現は取れるが、

酔いがさめている(完了を表わしている)

という表現は誤りである。アスペクト素性については6.3節で解説を行なっている。

「判断」の部分には「ない(否定)」や「らしい(推定)」「たい(希望)」といった機能を表わす助動詞がくる。「判断の時制」ということで助動詞の部分の時制が規定されることもある。

最後の「働きかけ」の部分には、「命令」「疑問」「禁止」といった終助詞を初めとする助詞で表現される語がくる。助詞や助動詞といった付属語は、前に来る語の品詞や活用形を規定するので、述語の生成は語尾の方から行なわなければならない。活用形の規定は、～行の～段活用の～形というだけでなく、音便形の選択などの問題も考慮しなければならない。音便形の選択を考慮しないと、

彼は釣りた魚を近所に配りた

といったおかしな日本語文が生成されてしまう。

以上述べた点を考慮した述語生成システムが**図6-6**に挙げる辞書および**図6-7**(p.174)のプログラムである。

述語の生成だけで実験したい場合は、

?-述語生成(_辞書,_命題,_アスペクト,T1,Mode,T2,Func).

を実行すればよい。_辞書の部分は6.3節で述べた動詞辞書の内容が入る。_命題の部分は

[]
受身
可能
使役

の3つのうちのどれかである。以下共通だがデフォールトは[](ヌルリスト)である。_アスペクトの部分には

[]
未完了(アスペクト素性が[+,+]、[+,−]、[−,+]の時)
完了
未完了反復(アスペクト素性が[−,−]の時)
未完了結果(アスペクト素性が[−,−]の時)

(p.173へ続く)

```
%****************%
%        か５    %
%        咲く    %
%****************%
か５([-320871_1],_1,未然).
か５([-320791_1],_1,未然５).
か５([-320851_1],_1,連用).
か５([-320631_1],_1,連用音便).
か５([-320941_1],_1,連用音便).
か５([-320831_1],_1,終止).
か５([-320831_1],_1,連体).
か５([-320941_1],_1,連体).
か５([-320811_1],_1,仮定).
か５([-320811_1],_1,命令).
%****************%
%        が５    %
%        騒ぐ    %
%****************%
が５([-320861_1],_1,未然).
が５([-320781_1],_1,未然５).
が５([-320841_1],_1,連用).
が５([-320941_1],_1,連用音便).
が５([-320821_1],_1,終止).
が５([-320821_1],_1,連体).
が５([-320801_1],_1,仮定).
が５([-320801_1],_1,命令).
%****************%
%        さ５    %
%        貸す    %
%****************%
さ５([-320771_1],_1,未然).
さ５([-320691_1],_1,未然５).
さ５([-320751_1],_1,連用).
さ５([-320731_1],_1,終止).
さ５([-320731_1],_1,連体).
さ５([-320711_1],_1,仮定).
さ５([-320711_1],_1,命令).
%****************%
%        た５    %
%        立つ    %
%****************%
た５([-320671_1],_1,未然).
た５([-320581_1],_1,未然５).
た５([-320651_1],_1,連用).
た５([-320631_1],_1,連用音便).
た５([-320621_1],_1,終止).
た５([-320621_1],_1,連体).
た５([-320601_1],_1,仮定).
た５([-320601_1],_1,命令).
```

```
%****************%
%       な５     %
%       死ぬ     %
%****************%
な５([-320561_1],_1,未然).
な５([-320521_1],_1,未然５).
な５([-320551_1],_1,連用).
な５([-320151_1],_1,連用音便).
な５([-320541_1],_1,終止).
な５([-320541_1],_1,連体).
な５([-320531_1],_1,仮定).
な５([-320531_1],_1,命令).
%****************%
%       ば５     %
%       飛ぶ     %
%****************%
ば５([-320501_1],_1,未然).
ば５([-320381_1],_1,未然５).
ば５([-320471_1],_1,連用).
ば５([-320151_1],_1,連用音便).
ば５([-320441_1],_1,終止).
ば５([-320441_1],_1,連体).
ば５([-320411_1],_1,仮定).
ば５([-320411_1],_1,命令).
%****************%
%       ま５     %
%       飲む     %
%****************%
ま５([-320361_1],_1,未然).
ま５([-320321_1],_1,未然５).
ま５([-320351_1],_1,連用).
ま５([-320151_1],_1,連用音便).
ま５([-320341_1],_1,終止).
ま５([-320341_1],_1,連体).
ま５([-320331_1],_1,仮定).
ま５([-320331_1],_1,命令).
%****************%
%       ら５     %
%       乗る     %
%****************%
ら５([-320251_1],_1,未然).
ら５([-320211_1],_1,未然５).
ら５([-320241_1],_1,連用).
ら５([-320631_1],_1,連用音便).
ら５([-320231_1],_1,終止).
ら５([-320231_1],_1,連体).
ら５([-320221_1],_1,仮定).
ら５([-320221_1],_1,命令).
```

```
%*****************%
%        わあ５   %
%        思う     %
%*****************%
わあ５([-32019|_1],_1,未然).
わあ５([-32088|_1],_1,未然５).
わあ５([-32094|_1],_1,連用).
わあ５([-32063|_1],_1,連用音便).
わあ５([-32092|_1],_1,終止).
わあ５([-32092|_1],_1,連体).
わあ５([-32090|_1],_1,仮定).
わあ５([-32090|_1],_1,命令).
%*****************%
%        あ下一   %
%        考える   %
%*****************%
あ下一([-32090|_1],_1,未然).
あ下一([-32090|_1],_1,連用).
あ下一([-32090,-32023|_1],_1,終止).
あ下一([-32090,-32023|_1],_1,連体).
あ下一([-32090,-32022|_1],_1,仮定).
あ下一([-32090,-32021|_1],_1,命令).
あ下一([-32090,-32026|_1],_1,命令).
%*****************%
%        か下一   %
%        助ける   %
%*****************%
か下一([-32081|_1],_1,未然).
か下一([-32081|_1],_1,連用).
か下一([-32081,-32023|_1],_1,終止).
か下一([-32081,-32023|_1],_1,連体).
か下一([-32081,-32022|_1],_1,仮定).
か下一([-32081,-32021|_1],_1,命令).
か下一([-32081,-32026|_1],_1,命令).
%*****************%
%        が下一   %
%        投げる   %
%*****************%
が下一([-32080|_1],_1,未然).
が下一([-32080|_1],_1,連用).
が下一([-32080,-32023|_1],_1,終止).
が下一([-32080,-32023|_1],_1,連体).
が下一([-32080,-32022|_1],_1,仮定).
が下一([-32080,-32021|_1],_1,命令).
が下一([-32080,-32026|_1],_1,命令).
```

```
%%%%%%%%%%%%%%%%%
%　　さ下一　%
%　　任せる　%
%%%%%%%%%%%%%%%%%
さ下一([-32071|_1],_1,未然).
さ下一([-32071|_1],_1,連用).
さ下一([-32071,-32023|_1],_1,終止).
さ下一([-32071,-32023|_1],_1,連体).
さ下一([-32071,-32022|_1],_1,仮定).
さ下一([-32071,-32021|_1],_1,命令).
さ下一([-32071,-32026|_1],_1,命令).
%%%%%%%%%%%%%%%%%
%　　ざ下一　%
%　　混ぜる　%
%%%%%%%%%%%%%%%%%
ざ下一([-32070|_1],_1,未然).
ざ下一([-32070|_1],_1,連用).
ざ下一([-32070,-32023|_1],_1,終止).
ざ下一([-32070,-32023|_1],_1,連体).
ざ下一([-32070,-32022|_1],_1,仮定).
ざ下一([-32070,-32021|_1],_1,命令).
ざ下一([-32070,-32026|_1],_1,命令).
%%%%%%%%%%%%%%%%%
%　　た下一　%
%　　立てる　%
%%%%%%%%%%%%%%%%%
た下一([-32060|_1],_1,未然).
た下一([-32060|_1],_1,連用).
た下一([-32060,-32023|_1],_1,終止).
た下一([-32060,-32023|_1],_1,連体).
た下一([-32060,-32022|_1],_1,仮定).
た下一([-32060,-32021|_1],_1,命令).
た下一([-32060,-32026|_1],_1,命令).
%%%%%%%%%%%%%%%%%
%　　だ下一　%
%　　出る　　%
%%%%%%%%%%%%%%%%%
だ下一([-32059|_1],_1,未然).
だ下一([-32059|_1],_1,連用).
だ下一([-32059,-32023|_1],_1,終止).
だ下一([-32059,-32023|_1],_1,連体).
だ下一([-32059,-32022|_1],_1,仮定).
だ下一([-32059,-32021|_1],_1,命令).
だ下一([-32059,-32026|_1],_1,命令).
```

```
%****************%
%        な下一   %
%        寝る     %
%****************%
な下一([-32053|_1],_1,未然).
な下一([-32053|_1],_1,連用).
な下一([-32053,-32023|_1],_1,終止).
な下一([-32053,-32023|_1],_1,連体).
な下一([-32053,-32022|_1],_1,仮定).
な下一([-32053,-32021|_1],_1,命令).
な下一([-32053,-32026|_1],_1,命令).
%****************%
%        は下一   %
%        経る     %
%****************%
は下一([-32042|_1],_1,未然).
は下一([-32042|_1],_1,連用).
は下一([-32042,-32023|_1],_1,終止).
は下一([-32042,-32023|_1],_1,連体).
は下一([-32042,-32022|_1],_1,仮定).
は下一([-32042,-32021|_1],_1,命令).
は下一([-32042,-32026|_1],_1,命令).
%****************%
%        ば下一   %
%        並べる   %
%****************%
ば下一([-32041|_1],_1,未然).
ば下一([-32041|_1],_1,連用).
ば下一([-32041,-32023|_1],_1,終止).
ば下一([-32041,-32023|_1],_1,連体).
ば下一([-32041,-32022|_1],_1,仮定).
ば下一([-32041,-32021|_1],_1,命令).
ば下一([-32041,-32026|_1],_1,命令).
%****************%
%        ま下一   %
%        攻める   %
%****************%
ま下一([-32033|_1],_1,未然).
ま下一([-32033|_1],_1,連用).
ま下一([-32033,-32023|_1],_1,終止).
ま下一([-32033,-32023|_1],_1,連体).
ま下一([-32033,-32022|_1],_1,仮定).
ま下一([-32033,-32021|_1],_1,命令).
ま下一([-32033,-32026|_1],_1,命令).
```

```
%****************%
%        ら下一  %
%        流れる  %
%****************%
ら下一([-32022|_1],_1,未然).
ら下一([-32022|_1],_1,連用).
ら下一([-32022,-32023|_1],_1,終止).
ら下一([-32022,-32023|_1],_1,連体).
ら下一([-32022,-32022|_1],_1,仮定).
ら下一([-32022,-32021|_1],_1,命令).
ら下一([-32022,-32026|_1],_1,命令).
%****************%
%        あ上一  %
%        悔いる  %
%****************%
あ上一([-32094|_1],_1,未然).
あ上一([-32094|_1],_1,連用).
あ上一([-32094,-32023|_1],_1,終止).
あ上一([-32094,-32023|_1],_1,連体).
あ上一([-32094,-32022|_1],_1,仮定).
あ上一([-32094,-32021|_1],_1,命令).
あ上一([-32094,-32026|_1],_1,命令).
%****************%
%        か      %
%        起きる  %
%****************%
か上一([-32085|_1],_1,未然).
か上一([-32085|_1],_1,連用).
か上一([-32085,-32023|_1],_1,終止).
か上一([-32085,-32023|_1],_1,連体).
か上一([-32085,-32022|_1],_1,仮定).
か上一([-32085,-32021|_1],_1,命令).
か上一([-32085,-32026|_1],_1,命令).
%****************%
%        が上一  %
%        過ぎる  %
%****************%
が上一([-32084|_1],_1,未然).
が上一([-32084|_1],_1,連用).
が上一([-32084,-32023|_1],_1,終止).
が上一([-32084,-32023|_1],_1,連体).
が上一([-32084,-32022|_1],_1,仮定).
が上一([-32084,-32021|_1],_1,命令).
が上一([-32084,-32026|_1],_1,命令).
```

```
%****************%
%        ざ上一  %
%        恥じる  %
%****************%
ざ上一([-32074|_1],_1,未然).
ざ上一([-32074|_1],_1,連用).
ざ上一([-32074,-32023|_1],_1,終止).
ざ上一([-32074,-32023|_1],_1,連体).
ざ上一([-32074,-32022|_1],_1,仮定).
ざ上一([-32074,-32021|_1],_1,命令).
ざ上一([-32074,-32026|_1],_1,命令).
%****************%
%        た上一  %
%        落ちる  %
%****************%
た上一([-32065|_1],_1,未然).
た上一([-32065|_1],_1,連用).
た上一([-32065,-32023|_1],_1,終止).
た上一([-32065,-32023|_1],_1,連体).
た上一([-32065,-32022|_1],_1,仮定).
た上一([-32065,-32021|_1],_1,命令).
た上一([-32065,-32026|_1],_1,命令).
%****************%
%        な上一  %
%        煮る    %
%****************%
な上一([-32055|_1],_1,未然).
な上一([-32055|_1],_1,連用).
な上一([-32055,-32023|_1],_1,終止).
な上一([-32055,-32023|_1],_1,連体).
な上一([-32055,-32022|_1],_1,仮定).
な上一([-32055,-32021|_1],_1,命令).
な上一([-32055,-32026|_1],_1,命令).
%****************%
%        は上一  %
%        干る    %
%****************%
は上一([-32048|_1],_1,未然).
は上一([-32048|_1],_1,連用).
は上一([-32048,-32023|_1],_1,終止).
は上一([-32048,-32023|_1],_1,連体).
は上一([-32048,-32022|_1],_1,仮定).
は上一([-32048,-32021|_1],_1,命令).
は上一([-32048,-32026|_1],_1,命令).
```

```
%****************%
%        ば上一   %
%        延びる   %
%****************%
ば上一([-32047|_1],_1,未然).
ば上一([-32047|_1],_1,連用).
ば上一([-32047,-32023|_1],_1,終止).
ば上一([-32047,-32023|_1],_1,連体).
ば上一([-32047,-32022|_1],_1,仮定).
ば上一([-32047,-32021|_1],_1,命令).
ば上一([-32047,-32026|_1],_1,命令).
%****************%
%        ま上一   %
%        試みる   %
%****************%
ま上一([-32035|_1],_1,未然).
ま上一([-32035|_1],_1,連用).
ま上一([-32035,-32023|_1],_1,終止).
ま上一([-32035,-32023|_1],_1,連体).
ま上一([-32035,-32022|_1],_1,仮定).
ま上一([-32035,-32021|_1],_1,命令).
ま上一([-32035,-32026|_1],_1,命令).
%****************%
%        ら上一   %
%        懲りる   %
%****************%
ら上一([-32024|_1],_1,未然).
ら上一([-32024|_1],_1,連用).
ら上一([-32024,-32023|_1],_1,終止).
ら上一([-32024,-32023|_1],_1,連体).
ら上一([-32024,-32022|_1],_1,仮定).
ら上一([-32024,-32021|_1],_1,命令).
ら上一([-32024,-32026|_1],_1,命令).
%****************%
%        さ変     %
%****************%
さ変([-32077|_1],_1,未然).
さ変([-32075|_1],_1,未然).
さ変([-32075|_1],_1,連用).
さ変([-32073,-32023|_1],_1,終止).
さ変([-32073,-32023|_1],_1,連体).
さ変([-32073,-32022|_1],_1,仮定).
さ変([-32071,-32026|_1],_1,命令).
さ変([-32075,-32021|_1],_1,命令).
%****************%
%        か変     %
%****************%
か変([-32079|_1],_1,未然).
か変([-32085|_1],_1,連用).
か変([-32083,-32023|_1],_1,終止).
か変([-32083,-32023|_1],_1,連体).
か変([-32083,-32022|_1],_1,仮定).
か変([-32079,-32094|_1],_1,命令).
```

```
%************************%
%        形容詞語尾        %
%************************%
形容詞語尾([-32087,-32021|_1],_1,未然).
形容詞語尾([-32083|_1],_1,連用).
形容詞語尾([-32087,-32063|_1],_1,連用音便).
形容詞語尾([-32094|_1],_1,終止).
形容詞語尾([-32094|_1],_1,連体).
形容詞語尾([-32081,-32022|_1],_1,仮定).
%************************%
%        形容動詞語尾      %
%************************%
形容動詞語尾([-32066,-32021|_1],_1,未然).
形容動詞語尾([-32066,-32063|_1],_1,連用音便).
形容動詞語尾([-32059|_1],_1,連用).
形容動詞語尾([-32055|_1],_1,連用).
形容動詞語尾([-32066|_1],_1,終止).
形容動詞語尾([-32056|_1],_1,連体).
形容動詞語尾([-32056,-32025|_1],_1,仮定).
%****************%
%        使役      %
%****************%
せる([-32071|_1],_1,未然).
せる([-32071|_1],_1,連用).
せる([-32071,-32023|_1],_1,終止).
せる([-32071,-32023|_1],_1,連体).
せる([-32071,-32021|_1],_1,命令).
せる([-32071,-32026|_1],_1,命令).
させる([-32077,-32071|_1],_1,未然).
させる([-32077,-32071|_1],_1,連用).
させる([-32077,-32071,-32023|_1],_1,終止).
させる([-32077,-32071,-32023|_1],_1,連体).
させる([-32077,-32071,-32021|_1],_1,命令).
させる([-32077,-32071,-32026|_1],_1,命令).
%************************%
%        受身可能          %
%************************%
れる([-32022|_1],_1,未然).
れる([-32022|_1],_1,連用).
れる([-32022,-32023|_1],_1,終止).
れる([-32022,-32023|_1],_1,連体).
れる([-32022,-32022|_1],_1,仮定).
れる([-32022,-32021|_1],_1,命令).
れる([-32022,-32026|_1],_1,命令).
られる([-32025,-32022|_1],_1,未然).
られる([-32025,-32022|_1],_1,連用).
られる([-32025,-32022,-32023|_1],_1,終止).
られる([-32025,-32022,-32023|_1],_1,連体).
られる([-32025,-32022,-32022|_1],_1,仮定).
られる([-32025,-32022,-32021|_1],_1,命令).
られる([-32025,-32022,-32026|_1],_1,命令).
```

```
%************************%
%　　打ち消し　　%
%************************%
ない([-32056,-32087,-32021|_1],_1,未然).
ない([-32056,-32083|_1],_1,連用).
ない([-32056,-32087,-32063|_1],_1,連用音便).
ない([-32056,-32094|_1],_1,終止).
ない([-32056,-32094|_1],_1,連体).
ない([-32056,-32081,-32022|_1],_1,仮定).
%****************%
%　　希望　　%
%****************%
たい([-32067,-32087,-32021|_1],_1,未然).
たい([-32067,-32083|_1],_1,連用).
たい([-32067,-32087,-32063|_1],_1,連用音便).
たい([-32067,-32094|_1],_1,終止).
たい([-32067,-32094|_1],_1,連体).
たい([-32067,-32081,-32022|_1],_1,仮定).
%****************%
%　　推定　　%
%****************%
らしい([-32025,-32075,-32083|_1],_1,連用).
らしい([-32067,-32075,-32087,-32063|_1],_1,連用音便).
らしい([-32025,-32075,-32094|_1],_1,終止).
らしい([-32025,-32075,-32094|_1],_1,連体).
らしい([-32025,-32075,-32081,-32022|_1],_1,仮定).
%****************%
%　　様態　　%
%****************%
そうだ([-32069,-32092,-32066,-32021|_1],_1,未然).
そうだ([-32069,-32092,-32059|_1],_1,連用).
そうだ([-32069,-32092,-32055|_1],_1,連用).
そうだ([-32069,-32092,-32066,-32063|_1],_1,連用音便).
そうだ([-32069,-32092,-32066|_1],_1,終止).
そうだ([-32069,-32092,-32056|_1],_1,連体).
そうだ([-32069,-32092,-32056,-32025|_1],_1,仮定).
%****************%
%　　不確定　%
%****************%
ようだ([-32026,-32092,-32066,-32021|_1],_1,未然).
ようだ([-32026,-32092,-32059|_1],_1,連用).
ようだ([-32026,-32092,-32055|_1],_1,連用).
ようだ([-32026,-32092,-32066,-32063|_1],_1,連用音便).
ようだ([-32026,-32092,-32066|_1],_1,終止).
ようだ([-32026,-32092,-32056|_1],_1,連体).
ようだ([-32026,-32092,-32056,-32025|_1],_1,仮定).
```

```
%****************%
%        丁寧     %
%****************%
ます([-32036,-32071|_1],_1,未然).
ます([-32036,-32075,-32027|_1],_1,未然).
ます([-32036,-32075|_1],_1,連用).
ます([-32036,-32073|_1],_1,終止).
ます([-32036,-32073|_1],_1,連体).
ます([-32036,-32073,-32022|_1],_1,仮定).
ます([-32036,-32075|_1],_1,命令).
ます([-32036,-32071|_1],_1,命令).
%****************%
%        意志     %
%****************%
う([-32092|_1],_1,終止).
よう([-32026,-32092|_1],_1,終止).
%************************%
%        打ち消し意志     %
%************************%
まい([-32036,-32094|_1],_1,終止).
%************************%
%                        %
%        命題変換処理     %
%                        %
%************************%
%****************%
%        []      %
%****************%
命題変換(_3,_3,[],[_1,_2,Kt],[_1,_2,Kt]).
%****************%
%        受身     %
%****************%
命題変換(_2,_1,受身,[さ,変,未然],[φ,下一,Kt]) :-
    れる(_2,_1,Kt).
命題変換(_2,_1,受身,[_3,ら,未然],[φ,下一,Kt]) :-
    れる(_2,_1,Kt).
命題変換(_2,_1,受身,[か,変,未然],[φ,下一,Kt]) :-
    られる(_2,_1,Kt).
命題変換(_2,_1,受身,[_3,下一,未然],[φ,下一,Kt]) :-
    られる(_2,_1,Kt).
命題変換(_2,_1,受身,[_3,上一,未然],[φ,下一,Kt]) :-
    られる(_2,_1,Kt).
```

```
%*****************%
%        可能      %
%*****************%
命題変換(_2,_1,可能,[さ,変,未然],[φ,下一,Kt]) :-
   れる(_2,_1,Kt).
命題変換(_2,_1,可能,[_3,5,未然],[φ,下一,Kt]) :-
   れる(_2,_1,Kt).
命題変換(_2,_1,可能,[か,変,未然],[φ,下一,Kt]) :-
   られる(_2,_1,Kt).
命題変換(_2,_1,可能,[_3,下一,未然],[φ,下一,Kt]) :-
   られる(_2,_1,Kt).
命題変換(_2,_1,可能,[_3,上一,未然],[φ,下一,Kt]) :-
   られる(_2,_1,Kt).
%*****************%
%        使役      %
%*****************%
命題変換(_2,_1,使役,[さ,変,未然],[φ,下一,Kt]) :-
   せる(_2,_1,Kt).
命題変換(_2,_1,使役,[_3,5,未然],[φ,下一,Kt]) :-
   せる(_2,_1,Kt).
命題変換(_2,_1,使役,[か,変,未然],[φ,下一,Kt]) :-
   させる(_2,_1,Kt).
命題変換(_2,_1,使役,[_3,下一,未然],[φ,下一,Kt]) :-
   させる(_2,_1,Kt).
命題変換(_2,_1,使役,[_3,上一,未然],[φ,下一,Kt]) :-
   させる(_2,_1,Kt).
%*********************************%
%                                 %
%        アスペクト & 時制        %
%                                 %
%*********************************%
%*************************%
%        +状態,+継続      %
%*************************%
アスペクト(_2,_1,[+,+],[],過去,[が,5,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[+,+],[],過去,[な,5,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[+,+],[],過去,[ば,5,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[+,+],[],過去,[ま,5,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[+,+],[],過去,[_4,_5,_3],[φ,φ,Kt]) :-
   (_3=連用音便;_3=連用),
   た(_2,_1,Kt).
アスペクト(_1,_1,[+,+],[],[],Kt,Kt).
```

```
%************************%
%        ＋状態，－継続    %
%************************%
アスペクト(_2,_1,[+,-],未完了,過去,[が,５,連用音便],[φ,φ,Kt]) :-
   でいた(_2,_1,Kt).
アスペクト(_2,_1,[+,-],未完了,過去,[な,５,連用音便],[φ,φ,Kt]) :-
   でいた(_2,_1,Kt).
アスペクト(_2,_1,[+,-],未完了,過去,[ば,５,連用音便],[φ,φ,Kt]) :-
   でいた(_2,_1,Kt).
アスペクト(_2,_1,[+,-],未完了,過去,[ま,５,連用音便],[φ,φ,Kt]) :-
   でいた(_2,_1,Kt).
アスペクト(_2,_1,[+,-],未完了,過去,[_4,_5,_3],[φ,φ,Kt]) :-
   (_3=連用音便;_3=連用),
   ていた(_2,_1,Kt).
アスペクト(_2,_1,[+,-],未完了,[],[が,５,連用音便],[φ,φ,Kt]) :-
   でいる(_2,_1,Kt).
アスペクト(_2,_1,[+,-],未完了,[],[な,５,連用音便],[φ,φ,Kt]) :-
   でいる(_2,_1,Kt).
アスペクト(_2,_1,[+,-],未完了,[],[ば,５,連用音便],[φ,φ,Kt]) :-
   でいる(_2,_1,Kt).
アスペクト(_2,_1,[+,-],未完了,[],[ま,５,連用音便],[φ,φ,Kt]) :-
   でいる(_2,_1,Kt).
アスペクト(_2,_1,[+,-],未完了,[],[_4,_5,_3],[φ,φ,Kt]) :-
   (_3=連用音便;_3=連用),
   ている(_2,_1,Kt).
アスペクト(_1,_1,[+,-],[],[],Kt,Kt).
%************************%
%        －状態，－継続    %
%************************%

アスペクト(_2,_1,[-,-],未完了反復,過去,[_3,連用],[φ,Kt]) :-
   つつあった(_2,_1,Kt).
アスペクト(_2,_1,[-,-],未完了反復,[],[_3,連用],[φ,Kt]) :-
   つつある(_2,_1,Kt).
アスペクト(_2,_1,[-,-],未完了結果,過去,[が,５,連用音便],[φ,φ,Kt]) :-
   でいた(_2,_1,Kt).
アスペクト(_2,_1,[-,-],未完了結果,過去,[な,５,連用音便],[φ,φ,Kt]) :-
   でいた(_2,_1,Kt).
アスペクト(_2,_1,[-,-],未完了結果,過去,[ば,５,連用音便],[φ,φ,Kt]) :-
   でいた(_2,_1,Kt).
アスペクト(_2,_1,[-,-],未完了結果,過去,[ま,５,連用音便],[φ,φ,Kt]) :-
   でいた(_2,_1,Kt).
アスペクト(_2,_1,[-,-],未完了結果,過去,[_4,_5,_3],[φ,φ,Kt]) :-
   (_3=連用音便;_3=連用),
   ていた(_2,_1,Kt).
アスペクト(_2,_1,[-,-],未完了結果,[],[が,５,連用音便],[φ,φ,Kt]) :-
   でいる(_2,_1,Kt).
アスペクト(_2,_1,[-,-],未完了結果,[],[な,５,連用音便],[φ,φ,Kt]) :-
   でいる(_2,_1,Kt).
アスペクト(_2,_1,[-,-],未完了結果,[],[ば,５,連用音便],[φ,φ,Kt]) :-
   でいる(_2,_1,Kt).
```

```
アスペクト(_2,_1,[-,-],未完了結果,[],[ま,５,連用音便],[φ,φ,Kt]) :-
   でいる(_2,_1,Kt).
アスペクト(_2,_1,[-,-],未完了結果,[],[_4,_5,_3],[φ,φ,Kt]) :-
   (_3=連用音便;_3=連用),
   ている(_2,_1,Kt).
アスペクト(_2,_1,[-,-],完了,過去,[が,５,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,-],完了,過去,[な,５,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,-],完了,過去,[ば,５,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,-],完了,過去,[ま,５,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,-],完了,過去,[_4,_5,_3],[φ,φ,Kt]) :-
   (_3=連用音便;_3=連用),
   た(_2,_1,Kt).
アスペクト(_2,_1,[-,-],完了,[],[が,５,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,-],完了,[],[な,５,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,-],完了,[],[ば,５,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,-],完了,[],[ま,５,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,-];完了,[],[_4,_5,_3],[φ,φ,Kt]) :-
   (_3=連用音便;_3=連用),
   た(_2,_1,Kt).
アスペクト(_2,_1,[-,-],[],過去,[が,５,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,-],[],過去,[な,５,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,-],[],過去,[ば,５,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,-],[],過去,[ま,５,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,-],[],過去,[_4,_5,_3],[φ,φ,Kt]) :-
   (_3=連用音便;_3=連用),
   た(_2,_1,Kt).
アスペクト(_1,_1,[-,-],[],[],Kt,Kt).
```

```
%************************%
%        －状態，＋継続    %
%************************%
アスペクト(_2,_1,[-,+],未完了,過去,[が,５,連用音便],[φ,φ,Kt]) :-
    でいた(_2,_1,Kt).
アスペクト(_2,_1,[-,+],未完了,過去,[な,５,連用音便],[φ,φ,Kt]) :-
    でいた(_2,_1,Kt).
アスペクト(_2,_1,[-,+],未完了,過去,[ば,５,連用音便],[φ,φ,Kt]) :-
    でいた(_2,_1,Kt).
アスペクト(_2,_1,[-,+],未完了,過去,[ま,５,連用音便],[φ,φ,Kt]) :-
    でいた(_2,_1,Kt).
アスペクト(_2,_1,[-,+],未完了,過去,[_4,_5,_3],[φ,φ,Kt]) :-
    (_3=連用音便;_3=連用),
    ていた(_2,_1,Kt).
アスペクト(_2,_1,[-,+],未完了,[],[が,５,連用音便],[φ,φ,Kt]) :-
    でいる(_2,_1,Kt).
アスペクト(_2,_1,[-,+],未完了,[],[な,５,連用音便],[φ,φ,Kt]) :-
    でいる(_2,_1,Kt).
アスペクト(_2,_1,[-,+],未完了,[],[ば,５,連用音便],[φ,φ,Kt]) :-
    でいる(_2,_1,Kt).
アスペクト(_2,_1,[-,+],未完了,[],[ま,５,連用音便],[φ,φ,Kt]) :-
    でいる(_2,_1,Kt).
アスペクト(_2,_1,[-,+],未完了,[],[_4,_5,_3],[φ,φ,Kt]) :-
    (_3=連用音便;_3=連用),
    ている(_2,_1,Kt).
アスペクト(_2,_1,[-,+],完了,過去,[が,５,連用音便],[φ,φ,Kt]) :-
    だ(_2,_1,Kt).
アスペクト(_2,_1,[-,+],完了,過去,[な,５,連用音便],[φ,φ,Kt]) :-
    だ(_2,_1,Kt).
アスペクト(_2,_1,[-,+],完了,過去,[ば,５,連用音便],[φ,φ,Kt]) :-
    だ(_2,_1,Kt).
アスペクト(_2,_1,[-,+],完了,過去,[ま,５,連用音便],[φ,φ,Kt]) :-
    だ(_2,_1,Kt).
アスペクト(_2,_1,[-,+],完了,過去,[_4,_5,_3],[φ,φ,Kt]) :-
    (_3=連用音便;_3=連用),
    た(_2,_1,Kt).
アスペクト(_2,_1,[-,+],完了,[],[が,５,連用音便],[φ,φ,Kt]) :-
    だ(_2,_1,Kt).
アスペクト(_2,_1,[-,+],完了,[],[な,５,連用音便],[φ,φ,Kt]) :-
    だ(_2,_1,Kt).
アスペクト(_2,_1,[-,+],完了,[],[ば,５,連用音便],[φ,φ,Kt]) :-
    だ(_2,_1,Kt).
アスペクト(_2,_1,[-,+],完了,[],[ま,５,連用音便],[φ,φ,Kt]) :-
    だ(_2,_1,Kt).
アスペクト(_2,_1,[-,+],完了,[],[_4,_5,_3],[φ,φ,Kt]) :-
    (_3=連用音便;_3=連用),
    た(_2,_1,Kt).
アスペクト(_2,_1,[-,+],[],過去,[が,５,連用音便],[φ,φ,Kt]) :-
    だ(_2,_1,Kt).
アスペクト(_2,_1,[-,+],[],過去,[な,５,連用音便],[φ,φ,Kt]) :-
    だ(_2,_1,Kt).
```

```
アスペクト(_2,_1,[-,+],[],過去,[ば,5,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,+],[],過去,[ま,5,連用音便],[φ,φ,Kt]) :-
   だ(_2,_1,Kt).
アスペクト(_2,_1,[-,+],[],過去,[_4,_5,_3],[φ,φ,Kt]) :-
   (_3=連用音便;_3=連用),
   た(_2,_1,Kt).
アスペクト(_1,_1,[-,+],[],[],Kt,Kt).
%************************%
%       アスペクト活用   %
%************************%
%****************%
%       た       %
%****************%
た([-32067,-32021|_1],_1,未然).
た(_1,_1,連用).
た([-32067|_1],_1,終止).
た([-32067|_1],_1,連体).
%****************%
%       だ       %
%****************%
だ([-32066,-32021|_1],_1,未然).
だ(_1,_1,連用).
だ([-32066|_1],_1,終止).
だ([-32066|_1],_1,連体).
%****************%
%       でいた   %
%****************%
でいた([-32059,-32094|_1],_1,未然).
でいた([-32059,-32094|_1],_1,連用).
でいた([-32059,-32094,-32067|_1],_1,終止).
でいた([-32059,-32094,-32067|_1],_1,連体).
%****************%
%       ていた   %
%****************%
ていた([-32060,-32094|_1],_1,未然).
ていた([-32060,-32094|_1],_1,連用).
ていた([-32060,-32094,-32067|_1],_1,終止).
ていた([-32060,-32094,-32067|_1],_1,連体).
%****************%
%       でいる   %
%****************%
でいる([-32059,-32094|_1],_1,未然).
でいる([-32059,-32094|_1],_1,連用).
でいる([-32059,-32094,-32023|_1],_1,終止).
でいる([-32059,-32094,-32023|_1],_1,連体).
%****************%
%       ている   %
%****************%
ている([-32060,-32094|_1],_1,未然).
ている([-32060,-32094|_1],_1,連用).
ている([-32060,-32094,-32023|_1],_1,終止).
ている([-32060,-32094,-32023|_1],_1,連体).
```

```
%************************%
%        つつあった        %
%************************%
つつあった([-32062,-32062|_1],_1,未然).
つつあった([-32062,-32062,-32096,-32024|_1],_1,連用音便).
つつあった([-32062,-32062,-32096,-32063|_1],_1,連用音便).
つつあった([-32062,-32062,-32096,-32063,-32067|_1],_1,終止).
つつあった([-32062,-32062,-32096,-32063,-32067|_1],_1,連体).
%************************%
%        つつある          %
%************************%
つつある([-32062,-32062|_1],_1,未然).
つつある([-32062,-32062,-32096,-32024|_1],_1,連用).
つつある([-32062,-32062,-32096,-32063|_1],_1,連用音便).
つつある([-32062,-32062,-32096,-32023|_1],_1,終止).
つつある([-32062,-32062,-32096,-32023|_1],_1,連体).
%************************%
%                        %
%        話者の判断        %
%                        %
%************************%
%****************%
%          []      %
%****************%
話者の判断(_1,_1,[],Kt,Kt).
話者の判断(_2,_1,打ち消し,[_3,_4,未然],[φ,φ,Kt]) :-
    ない(_2,_1,Kt).
話者の判断(_2,_1,希望,[_3,_4,連用],[φ,φ,Kt]) :-
    たい(_2,_1,Kt).
話者の判断(_2,_1,推定,[_3,_4,終止],[φ,φ,Kt]) :-
    らしい(_2,_1,Kt).
話者の判断(_2,_1,様態,[_3,_4,連用],[φ,φ,Kt]) :-
    そうだ(_2,_1,Kt).
話者の判断(_2,_1,伝聞,[_3,_4,終止],[φ,φ,Kt]) :-
    そうだ(_2,_1,Kt).
話者の判断(_2,_1,不確定,[_3,_4,連体],[φ,φ,Kt]) :-
    ようだ(_2,_1,Kt).
話者の判断(_2,_1,丁寧,[_3,_4,連用],[φ,φ,Kt]) :-
    ます(_2,_1,Kt).
話者の判断(_2,_1,意志,[_3,5,未然],[φ,φ,Kt]) :-
    う(_2,_1,Kt).
話者の判断(_2,_1,意志,[_3,_4,未然],[φ,φ,Kt]) :-
    よう(_2,_1,Kt).
話者の判断(_2,_1,打ち消し意志,[_3,5,終止],[φ,φ,Kt]) :-
    まい(_2,_1,Kt).
話者の判断(_2,_1,打ち消し意志,[_3,_4,未然],[φ,φ,Kt]) :-
    まい(_2,_1,Kt).
```

```
%************************%
%                        %
%        判断時制        %
%                        %
%************************%
判断時制(_2,_1,過去,_3,Kt) :-
   (_3=連用音便;_3=連用),
   た(_2,_1,Kt).
判断時制(_1,_1,[],Kt,Kt).
%************************%
%                        %
%        話者の働きかけ  %
%                        %
%************************%
話者の働きかけ([-32030,-320251_3],_3,連言,[_1,_2,連体]).
話者の働きかけ([-320871_3],_3,選言,[_1,_2,終止]).
話者の働きかけ([-32056,-320571_3],_3,例示,[_1,_2,連体]).
話者の働きかけ([-32083,-32025,-320941_3],_3,程度,[_1,_2,連体]).
話者の働きかけ([-32082,-32025,-320941_3],_3,程度,[_1,_2,連体]).
話者の働きかけ([-32036,-320591_3],_3,帰着点,[_1,_2,連体]).
話者の働きかけ([-32077,-320901_3],_3,類推,[_1,_2,連用]).
話者の働きかけ([-32059,-320321_3],_3,類推,[_1,_2,連用]).
話者の働きかけ([-32066,-32063,-320601_3],_3,類推,[_1,_2,連用]).
話者の働きかけ([-32075,-320871_2],_2,限定,[形容詞,_1,連用]).
話者の働きかけ([-32075,-320871_2],_2,限定,[形容動詞,_1,連用]).
話者の働きかけ([-32075,-320871_3],_3,限定,[_1,_2,連体]).
話者の働きかけ([-32050,-32087,-320241_3],_3,限定,[_1,_2,連体]).
話者の働きかけ([-32066,-320811_3],_3,限定,[_1,_2,連体]).
話者の働きかけ([-320501_3],_3,仮定,[_1,_2,仮定]).
話者の働きかけ([-320581_3],_3,仮定,[_1,_2,終止]).
話者の働きかけ([-32087,-320251_3],_3,原因,[_1,_2,終止]).
話者の働きかけ([-32052,-320591_3],_3,原因,[_1,_2,連体]).
話者の働きかけ([-320591_1],_1,順接,[が,５,連用]).
話者の働きかけ([-320591_1],_1,順接,[な,５,連用]).
話者の働きかけ([-320591_1],_1,順接,[ば,５,連用]).
話者の働きかけ([-320591_1],_1,順接,[ま,５,連用]).
話者の働きかけ([-320601_3],_3,順接,[_1,_2,連用]).
話者の働きかけ([-32066,-320241_1],_1,並列,[が,５,連用]).
話者の働きかけ([-32066,-320241_1],_1,並列,[な,５,連用]).
話者の働きかけ([-32066,-320241_1],_1,並列,[ば,５,連用]).
話者の働きかけ([-32066,-320241_1],_1,並列,[ま,５,連用]).
話者の働きかけ([-32067,-320241_3],_3,並列,[_1,_2,連用]).
話者の働きかけ([-32056,-32086,-320251_2],_2,並列,[形容詞,_1,終止]).
話者の働きかけ([-32056,-32086,-320251_3],_3,並列,[_1,_2,連用]).
話者の働きかけ([-32059,-320321_1],_1,逆接,[が,５,連用]).
話者の働きかけ([-32059,-320321_1],_1,逆接,[な,５,連用]).
話者の働きかけ([-32059,-320321_1],_1,逆接,[ば,５,連用]).
話者の働きかけ([-32059,-320321_1],_1,逆接,[ま,５,連用]).
話者の働きかけ([-32060,-320321_3],_3,逆接,[_1,_2,連用]).
話者の働きかけ([-32081,-32022,-320571_3],_3,逆接,[_1,_2,終止]).
話者の働きかけ([-32081,-32022,-32057,-320321_3],_3,逆接,[_1,_2,終止]).
話者の働きかけ([-320861_3],_3,逆接,[_1,_2,終止]).
```

```
話者の働きかけ([-32052,-32055|_3],_3,逆接,[_1,_2,連体]).
話者の働きかけ(_3,_3,命令,[_1,_2,命令]).
話者の働きかけ([-32060|_4],_4,命令,[_2,_3,_1]) :-
   (_1=連用音便;_1=連用).
話者の働きかけ([-32056,-32077,-32094|_4],_4,命令,[_2,_3,_1]) :-
   (_1=連用音便;_1=連用).
話者の働きかけ([-32087|_3],_3,疑問,[_1,_2,連体]).
話者の働きかけ([-32052|_3],_3,疑問,[_1,_2,連体]).
話者の働きかけ([-32056|_3],_3,禁止,[_1,_2,連体]).
話者の働きかけ([-32053|_3],_3,確認,[_1,_2,終止]).
話者の働きかけ(_3,_3,平叙,[_1,_2,終止]).
話者の働きかけ(_3,_3,[],[_1,_2,終止]).
```

図6-6　述語生成ルーチンの辞書

のうちのいずれかがはいる。T1は時制を表わし、

[]
過去

の2つのいずれかがはいる。[](ヌルリスト)は非過去(現在あるいは未来)を表わしている。アスペクト素性が[+,+]の場合の未完了過去、完了過去、未完了非過去、完了非過去、[+,−]の場合の完了過去、中立=[]には対応する表現はない事に注意。

Mode には6.3で述べた「話者の判断」を示すアトムがはいる。T2は話者の判断を示す助動詞の時制である。そこには動詞の時制と同様

[]
過去

のうちのどちらかがはいる。最後の Func には6.3節で述べた「話者の働きかけ」を示すアトムが入る。図6-8にいくつかの実行例を示す。

```
%************************%
%                        %
%        生成時          %
%        動詞命題の処理  %
%                        %
%************************%
述語生成([[S0,S1],[_行,_段],_,Asp],_命題,Osr,T1,Mode,T2,Func) :-
        命題生成(S0,[],[[S0,S1],[_行,_段],_,Asp],_命題,Osr,T1,Mode,T2,Fu
nc),
        knjprint(S0).
命題生成(S0,St,[[S0,S1],[_行,_段],_,Asp],_命題,Osr,T1,Mode,T2,Func) :-
        M6 =.. [話者の働きかけ,S6,St,Func,Kt5],
        M6,
        M5 =.. [判断時制,S5,S6,T2,Kt4,Kt5],
        M5,
        M4 =.. [話者の判断,S4,S5,Mode,Kt3,Kt4],
        M4,
        M3 =.. [アスペクト,S3,S4,Asp,Osr,T1,Kt2,Kt3],
        M3,
        M2 =.. [命題変換,S2,S3,_命題,[_行,_段,Kt1],Kt2],
        M2,
        活用形(_行,_段,_活用),
        M1 =.. [_活用,S1,S2,Kt1],
        M1.
活用形(_行,_段,_活用) :-
        name(_行,N1),
        name(_段,N2),
        append(N1,N2,N3),
        name(_活用,N3),
        !.
append([A|X],Y,[A|Z]) :-
        append(X,Y,Z).
append([],X,X).
jwrite([]) :-
        !.
jwrite([X|Y]) :-
        integer(X),
        knjput(X),
        jwrite(Y).
jwrite([X|Y]) :-
        write(X),
        jwrite(Y).
knjput(X) :-
        A is X / 256 + 255,
        B is X mod 256 + 256,
        put(A),put(B).
```

図6-7　述語生成ルーチンのプログラム

```
| ?-話す(Jisho),述語生成(Jisho,[],完了,過去,[],[],疑問).
話したか

yes

| ?-話す(Jisho),述語生成(Jisho,[],未完了,過去,[],[],疑問).
話していたか

yes

| ?-話す(Jisho),述語生成(Jisho,[],未完了,[],[],[],疑問).
話しているか

yes

| ?-作る(Jisho),述語生成(Jisho,使役,完了,[],推定,[],[]).
作らせたらしい

yes

| ?-作る(Jisho),述語生成(Jisho,[],完了,[],[],[],[]).
作った

yes

| ?-作る(Jisho),述語生成(Jisho,[],未完了,[],[],[],[]).
作っている

yes
```

図6-8　述語生成の実行例

6.4.2　日本語の生成

以下の2点に焦点を当てて、日本語文生成システムを構築する。

① 動詞、名詞、形容詞、副詞といった文を構成する要素をどういった語順で表現するか。

② 埋め込み文を処理するにはどうしたら良いか。

図6-9が意味ネットワークから日本語を生成するためのプログラムである。このプログラムと図6-3の動詞辞書、図6-4、図6-5の意味ネットワークおよび図6-6の述語生成の辞書、図6-7の述語生成のプログラムをコンサルトして

(p.179へ続く)

```
%************************%
%                        %
%        平叙文          %
%                        %
%************************%
日本語文(S1,St,Node,SubNode) :-
        手続き_日本語文(Node,SubNode,_辞書),
        述部(S3,St,Node,_辞書),
        副詞_時(S2,S3,Node,[]),
        主部(S1,S2,Node,_辞書).
%**************************************%
%       手続き_日本語文(Node,_辞書)     %
%**************************************%
手続き_日本語文(Node,SubNode,[X,Y,Z,W]) :-
        M =.. [Node,[X,Y,Z1,W]],
        M,
        格束縛(Node,Z1,Z2),
        remove(SubNode,Z2,Z).
格束縛(_,[],[]).
格束縛(Node,[[_格,X,_格条件,_格内容]|T],[[_格,X,_格条件,_格内容]|Ret]) :
-
        link(_格,[述語,Node],_格内容),
        !,
        generic(_格内容,_格条件),
        格束縛(Node,T,Ret).
格束縛(Node,[_|T],Ret) :-
        格束縛(Node,T,Ret).
generic(X,X).
generic(_格内容,_格条件) :-
        link(isa,_格内容,X),
        generic(X,_格条件).
mark(X) :-
        marked(X),
        !,fail.
mark(X) :-
             asserta(marked(X)),
        !.
remove(_,[],[]).
remove(SubNode,[[_,_,_,SubNode]|T],T).
remove(SubNode,[X|T1],[X|T2]) :-
        remove(SubNode,T1,T2).
```

```
%********************************%
%        副詞(S1,S2,Node,[])     %
%********************************%
副詞_時([Time|S1],S1,Node,[]) :-
        link(time,[述語,Node],[_,Time]),
        副詞_場所(S1,St,Node,[]).
副詞_時(S1,St,Node,[]) :-
        副詞_場所(S1,St,Node,[]).
副詞_場所([Locate|S1],S1,Node,[]) :-
        link(locate,[述語,Node],[_,Locate]),
        副詞(S1,St,Node,[]).
副詞_場所(S1,St,Node,[]) :-
        副詞(S1,St,Node,[]).
副詞([Adv|St],St,Node,[]) :-
        link(adv,[述語,Node],[_,Adv]).
副詞(St,St,Node,[]).
%************************%
%                        %
%        述部            %
%                        %
%************************%
述部(S1,St,Node,_辞書) :-
        手続き_述部(Node,_命題,Osr,T1,Mode,T2,Func),
        命題生成(S1,St,_辞書,_命題,Osr,T1,Mode,T2,Func).
%************************************************************%
%        手続き_述部(Node,_命題,Osr,T1,Mode,T2,Func)     %
%************************************************************%
手続き_述部(Node,_命題,Osr,T1,Mode,T2,Func) :-
        (link(命題,[述語,Node],_命題);_命題 = []),
        (link(osr,[述語,Node],Osr);Osr = []),
        (link(time,[述語,Node],[T1,_]);T1 = []),
        (link(話者の判断,[述語,Node],[Mode,T2]);(Mode = [],T2 = [])),
        (link(話者の働きかけ,[述語,Node],Func);Func = []).
%************************%
%                        %
%        主部            %
%                        %
%************************%
主部(S1,St,Node,[_,_,_格構造,_]) :-
        主部1(S1,St,Node,_格構造).
主部1(St,St,Node,[]).
主部1(S1,St,Node,[[_格,_格助詞,_,_格内容]|T]) :-
        名詞句1(S1,S2,_格内容,_格助詞),
        主部1(S2,St,Node,T).
```

```
%************************%
%                        %
%        名詞句          %
%                        %
%************************%

%************************%
%        名詞句1         %
%************************%
名詞句1(S1,St,Node,_格助詞) :-
        link(_,[述語,Next_Node],Node),
        mark(Next_Node),
        日本語文(S1,S2,Next_Node,Node),
        名詞句2(S2,St,Node,_格助詞).
名詞句1(S1,St,Node,_格助詞) :-
        名詞句2(S1,St,Node,_格助詞).
%************************%
%        名詞句2         %
%************************%
名詞句2(S1,St,Node,_格助詞) :-
        link(owner,Node,Owner),
        名詞句(S1,S2,Owner,[]),
        名詞句3(S2,[の|St],Node,_格助詞).
名詞句2(S1,St,Node,_格助詞) :-
        名詞句3(S1,St,Node,_格助詞).
%************************%
%        名詞句3         %
%************************%
名詞句3(S1,St,Node,_格助詞) :-
        link(adj,Node,[_,Adj]),
        形容詞(S1,S2,Adj,[]),
        名詞句4(S2,St,Node,_格助詞).
名詞句3(S1,St,Node,_格助詞) :-
        名詞句4(S1,St,Node,_格助詞).
%****************%
%        名詞    %
%****************%
名詞句4([Node,_格助詞|St],St,Node,_格助詞).
%****************%
%                %
%        形容詞  %
%                %
%****************%
形容詞([Adv|S1],S1,Node,[]) :-
        link(adv,Node,[_,Adv]),
        形容詞1(S1,St,Node,[]).
形容詞(S1,St,Node,[]) :-
        形容詞1(S1,St,Node,[]).
形容詞1([Node|St],St,Node,[]).
ex(Node) :-
        abolish(marked,1),
```

```
link(_,[述語,Node],_),
mark(Node),
日本語文(X,[],Node,[]),
!,
jwrite(X),nl.
```

図6-9　日本語生成プログラム

?-ex (_述語ノード).

と入力する。述語ノードとしては「作る」「食べる」「話す」「来る」を入れることができる。例えば

?-ex (食べる).

と入力すると

哲夫が裕子が卵で昨日作った料理を一人で食べている

という文が生成され、

?-ex (作る).

と入力すると

裕子が哲夫が一人で食べている料理を卵で昨日作った

という文が生成される。

文を構成する動詞・名詞・形容詞・副詞といった様々な要素は、6.3節で見たように、意味ネットワークによって表現してある。日本語生成プログラムはこの意味ネットワークを参照しつつ、日本語の文法に従って各要素をつなげていく。プログラムの先頭は

日本語文(S1,St,Node,SubNode)

となっているが、それぞれの引き数は以下のような役割を持っている。

S1　　：生成された日本語文が入る
St　　：ヌルリスト[]が入る
Node　：生成を始める述語ノードを指定する
SubNode：最初はヌルリスト[]が入る。埋め込み文の処理を行なうときは、ここに主文の述語ノードが埋め込み文へ渡される

上の述語の中では

手続き_日本語文(Node,SubNode,_辞書),
述部(S3,St,Node,_辞書),
副詞_時(S2,S3,Node,[]),
主部(S1,S2,Node,_辞書).

といった一連の処理が行なわれる。手続き_日本語文は、動詞辞書の第3引き数にある格の制約に関する情報と実際の意味ネットワークのマッチングをとり、意味ネットワークの内容が格に関する制約を満たしていれば、実際の内容を束縛する。このマッチングは以下のような手順で行なわれる。

① 格の制約に関する情報を動詞辞書から取り出す。例えば「話す」の「object」は「生物」である。
② 意味ネットワーク中の「話す」の「object」を参照すると「先生」になっている。
③ 「先生」が「生物」であれば、「話す」の「object」を先生として束縛する。「先生」が「生物」であるかどうかは、意味ネットワーク中の「isa」リンクをたどっていけば分かる。

その後は文の語尾の方向から、述部、副詞、主部の順に文章を構成していく。述部の構成は意味ネットワークから、時制、アスペクトなどの文の属性に関する情報を取ってきて述語生成ルーチンに渡すことで行なう。文の属性に関する情報は

手続き_述部(Node,_命題,Osr,T1,Mode,T2,Func),

という Prolog 述語で行なう。主部を構成する名詞句の生成は以下の手順で行なわれる。

① その名詞句が他の動詞との共通の格になっている場合は、埋め込み文の処理を行なう。
② owner リンクがあれば、名詞句の所有者としてその内容を加える。
③ adj リンクがあれば、名詞句の修飾語としてその内容を加える。形容詞にかかる副詞の処理は形容詞の処理の時に行なわれる。
④ 格に応じた格助詞をつけて、名詞句を構成しつけ加える。

このうち埋め込み文の処理は、いちばん最初の日本語文/4を再帰的に呼び出すことで処理を行なう。この時は、2つの述語によって共有されている名詞句が、生成された文章に重複しないように埋め込み文の方からは取り除く処

理を行なう。そうしないと

哲夫が裕子が料理を昨日作った料理を一人で食べている

といったおかしな文章が生成されてしまう。埋め込み文の処理の時は、既に言及された動詞と、埋め込み文の動詞の区別をつけるためにmarked/1という述語で処理の終わった動詞にマークをつけるようにしている。

6.5　システムの評価

第5章で紹介した「小さな積木の世界」は、限定された領域でのを対象としているので、辞書の記述量もそう大きくはない。しかし、この章の文生成システムは、汎用の辞書を記述しているために、辞書の記述量はかなり大きくなっている。記述量が大きいことを除けば、システム構築にかかる手間はそれほど大きくはない。特に、述語生成システムさえしっかりしたものになっていれば、中間表現を意味ネットワーク以外のものに切り替えたり、この章では扱っていない、5w1h形式の疑問文や従位接続詞を含んだ複文に対応することも簡単にできる。

実行スピードについては、パソコン上のPrologでも十分実用的なスピードで動作する。ただし、辞書を拡張していった場合、容量の方がさきに天井を打つであろう。推論システムなどと組み合わせる場合は、大容量のメモリをサポートしている処理系を利用する方が良いだろう。

文生成システムの応用分野であるが、機械翻訳システムの一部としての利用や、コンサルテーションシステムやCAI(コンピュータ補助教育)システムの出力部への利用が考えられる。その他にも意味ネットワークという知識表現は推論システムの記述も十分可能であるために、いろいろと応用がきくであろう。

今回挙げた述語生成ルーチンは、「活用形が後に続く語に規定される」という用言の特性のために、語尾の方から生成するようになっている。従って、辞書とのマッチングも語尾の方から行なっており、そのまま構文解析に使おうとすると相当効率が悪くなる。また、文生成においては、意味表現(中間表現)の方に曖昧性がないので、辞書は解析に用いられるものよりも相当簡略化されている。例えば、命令を表わす語尾の場合、文生成の方では

～しろ

といった命令形の活用語尾を採用すれば事が足りる。しかし、文解析の方では

～しろ
～して
～しなさい
～しましょう

など様々の表現を同じ命令形として判断しなければならない。従って、この辞書を述語の解析に変更する場合は、同義語に対応できるように辞書の項目を拡張し、動詞から活用語尾の方にマッチングを取るようにする必要があるだろう。その際図6-6の述語生成辞書は、拡張していく際のベースとして、そのまま用いることができるだろう。