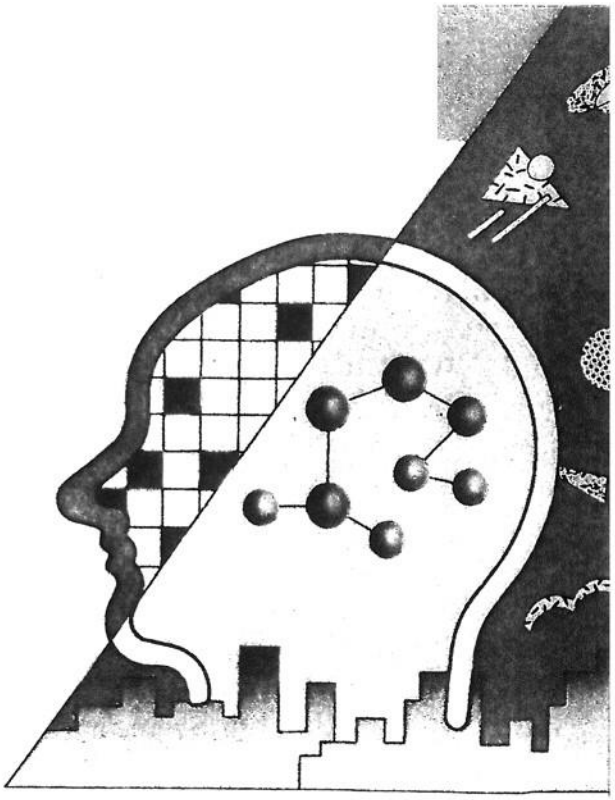
Prologによる機械翻訳システム その5

高野　真



Prologによる機械翻訳システムの連載は，今回と次回の2回で終了する。今回までシステムを完成させる予定であったが，都合により完成は最終回に延期する。今回は英語を構文解析し，中間表現に変換するまでの部分を紹介する。中間表現には格文法に近い表現を採用する。
最終回では，連載の第2回，第3回でインプリメントした日本語生成システムと英文の解析システムとの結合を行なう。

はじめに，構文解析システムが出力するべき目的表現である「中間表現」について解説する。次に，前回の形態素解析システムを含んだ構文解析システムをインプリメントする。インプリメントは，Prolog－KABAで行なった。

機械翻訳システムの中間表現

翻訳システムの中間表現に求められるものは，ふたつの言語の「ある状況における標準的な言語表出」に対応するすべての意味を表現する機能である。

意味表現に関する研究はAIにおいて20年来研究されているが，ある自然言語の表わし得る意味をすべて表現できるものはいまだに存在しない。しかし，「ある状況における標準的な言語表出」を表現することなら可能である。

機械翻訳システムの場合，「標準的な言語表出」とは，暗示的なニュアンスや比喩，曖昧な意味を含まない，伝達意図の明確な書き言葉に対応している。

今回採用する中間表現は，格文法に基づいた「命題表現」(Kintsch)を参考にしている。格文法は，動詞を中心とした単語の機能的な関係を表現するのに適した意味表現である。格文法での意味表現の単位になるのは，「命題」である。

たとえば，

「クラークケントはスーパーマンである」

「雨が降った」

「トムはキムを愛している」

は，それぞれひとつの命題を表わしている。

命題に対しては，真偽値を問うことができる。つまり，「YesかNoかを問うことのできる最小の単位」が命題ということになる。

命題は，要素と要素間の関係によって構成することができる。

上の例を命題形式で表現すると，

isa(クラークケント，スーパーマン)

降る(雨)

愛している(トム，キム)

のように表現することができる。

かっこの中のクラークケント，スーパーマン，雨，トム，キムが，命題を構成する要素に当たる。格文法では，状態や動作を表わす命題を構成する要素に「格(Case)」と呼ばれる役割を与えている。

かっこの前についた“isa”や“降る”，“愛している”が要素間の関係である。動詞やふたつの集合の包含関係を表わすisa(イズアと読

む)関係,ふたつの要素が同一であることを表わすreference関係,修飾・接続関係が代表的なものである。

命題は,要素として他の命題を取ることもできる。

たとえば,

今日雨が降った

トムはキムをとても愛している

は,

時(降る(雨),今日)

修飾(愛している(トム,キム),とても)

のように表現することができる。

次に,命題表現に基づいた表現法について解説する。命題の要素,動詞などの関係名については,中間表現になった時点ですでに日本語に翻訳されているものとする。

まず,命題のタイプを以下の3つに分ける。これはKintchの用いた分類法に準じている。

①Predication——状態や行為を表わす。要素間の関係としては動詞や"isa"が入る。

②Modification——形容詞,副詞による修飾,全体部分関係,否定関係も含む。

③Connection——接続関係,比較,時や場所に関する記述もここに入る。

まず最初に,①Predication命題について解説しよう。

この命題には,3通りのタイプがある。ひとつは,命題間の関係として動詞を持つ命題である。その場合命題の要素には,格が設定される。格文法を提唱したフィルモアは以下のような格を設定している。

①agent——行為や状態を引き起こした物(人)

②Experiencer——心理・感情の主体

③Instrument——行為や状態での道具,刺激

④Object——行為の対象

⑤Source——行為や状態の源

⑥Goal——行為や状態の結果,目的

しかし,これだけの格では表現しきれない要素もあり,実際に運用されるときは別の格を設定することが多い。

図1に,要素間の関係に動詞を取る命題の例を挙げよう。

```
Betty bougut a candy.
        買う(
                agent:ベティー,
                object:キャンディー
        ).

I give a present to Kim.
        与える(
                agent:私,
                object:プレゼント,
                goal:キム
        ).
```

図1　動詞を関係としてとる命題

```
Clark Kent is a Superman.
        isa(
                クラークケント,
                スーパーマン
        ).

Penguin is a bird.
        isa(
                ペンギン,
                鳥
        ).
```

図2　isa関係の例

Predication命題の残りふたつの関係には,"isa"と"reference"がある。"isa"関係は,ふたつの集合間の包含関係を表わし,"reference"関係はふたつの要素のイコール関係を表わす。

これらの関係はbe動詞で表わされることが多く,今回は両方とも"isa"で表わすことにしている。

図2に,"isa"関係の例を挙げる。

次に,②Modification関係について説明しよう。この関係には図3に例と共に挙げる4つのタイプがある。今回はいずれも"Modification"関係としてまとめて扱うことにする。また,BがAを修飾する事を表わす記述,

```
Modification(
        A,
        B
).
```

は,

```
B(
        A
).
```

というように,略記することもできる。

次に,③Connection関係について説明しよう。この関係には,図4に挙げる10のタイプがある。

Connection関係は,接続詞や前置詞によって明示的に表わされるものだけを扱う。

この関係の例を,図5に挙げる

```
②－１　定性的修飾
  定性的な性質について言及した修飾語

Tom is silly.
        modification(
                トム，
                馬鹿
        ).
        馬鹿(
                トム
        ).


②－２　定量的修飾
  定量的な性質について言及した修飾語

three baggs
        modification(
                バッグ，
                3
        ).
        3(
                バッグ
        ).

②－３　全体部分関係・所有関係
  ある物の一部や所有物であることを示す修飾語

pencil's cap
        modification(
                キャップ，
                鉛筆の
        ).
        鉛筆の(
                キャップ
        ).

Tom's bag
        modification(
                カバン，
                トムの
        ).
        トムの(
                カバン
        ).

②－４　否定関係

Kim doesn't play baseball.
        modification(
                する(
                        agent:トム，
                        object:野球
                ),
                negative
        ).
        negative(
                する(
                        agent:トム，
                        object:野球
                )
        ).
```

図３　Modification関係

```
③－１　ｌｏｃａｔｉｏｎ
  一つの命題が発生した場所に関する記述

③－２　時間係
  一つの命題が発生した時に関する記述

③－３　ｃｏｎｊｕｎｃｔｉｏｎ
  命題間のａｎｄ関係

③－４　ｄｉｓｊｕｎｃｔｉｏｎ
  命題間のｏｒ関係

③－５　ｃａｕｓａｌｉｔｙ
  原因結果関係を表わす

③－６　ｐｕｒｐｏｓｅ
  目的を表わす

③－７　ｃｏｎｓｅｓｓｉｏｎ
  「～にもかかわらず」に類する関係を表わす

③－８　Ｃｏｎｔｒａｓｔ
  比較級、最上級の表現に対応する関係

③－９　Ｃｏｎｄｉｔｉｏｎ
  「もし～したら」といった仮定関係

③－１０　Ｍａｎｎｅｒ
  様態を表わす
```

図４　Connection関係

```
Tom and Kim went to the store.
        行く(
                agent:conjunction(
                        トム，
                        キム
                ),
                goal:店
        ).

Tom goes to the beach to swim.
        purpose(
                行く(
                        agent:トム，
                        goal:海岸
                ),
                泳ぐ(
                        agent:トム，
                )
        ).

Kim was born in 1960.
        time(
                生まれる(
                        agent:キム
                ),
                １９６０年に
        ).

Tom skates like a professional.
        manner(
                スケートをする(
                        agent:トム
                ),
                プロのように
        ).
```

図５　Connection関係の例

Kintschの命題表現は，もともと複数の文からなる文章に適用するための意味表現の体系であったが，今回は他の文との関係についてはいっさい考慮を払っていない。実際には，接続詞などで明示的に示されないConnection関係が数多く存在することは，意識しておかなければならないだろう。

構文解析システム

構文解析システムの入力には，前回の形態素解析システムの出力が用いられる。構文解析システムの出力は，今回解説した中間表現である。

今回のシステムでは，文法記述として単一化文法の枠組みを採用している。単一化文法の文法規則は，従来の句構造文法に比べて非常に少なくなっている。それは，文法範疇を抽象化し，語彙に接続情報を分担させることによって実現されている。

単一化文法の概略については前回解説をしたので，今回は詳しくは述べない。今回

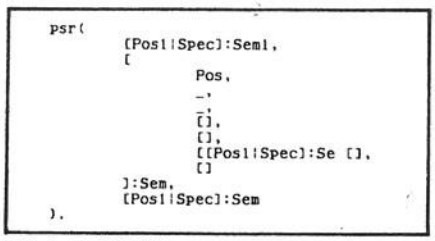


図6-1　文法規則 1

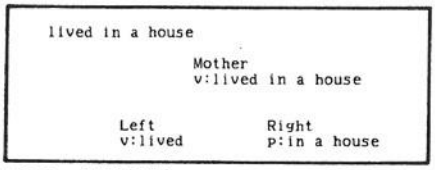


図6-2　文法規則 1

は，単一化文法で用いられている文法規則について詳しく見ていこう。

図6～図9に，今回用いた文法規則を挙げる。

"psr"とは"Phrase Structure Rule"の略である。述語psrの要素は3つで図6－2における文の要素に対応している。第1引き数は，図のLeft，第2引き数は図のRight，第3引き数は図のMotherである。

図6－2は，文法規則1を図で表わしたものである。文法規則1は，文の後の要素が前の要素を修飾する関係を表わしている。"lived in a house"の例の場合，文法カテゴリーが前置詞である"in a house"が，動詞の"lived"に付いて新たな動詞カテゴリーを生成することを表わしている。

各引き数の要素の意味については，前回に説明したとおりであるが，簡単に説明しよう。

リストの第1要素"Pos"は，品詞名を表わしている。今回は，

v　動詞
n　名詞
p　前置詞
a　形容詞，副詞
det　冠詞
junc　接続詞

のいずれかである。

第2要素と第3要素は，その品詞が原形か変化形か，格はなにかといった属性を表わしている。

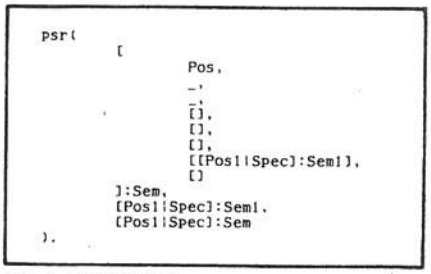


図7-1　文法規則 2

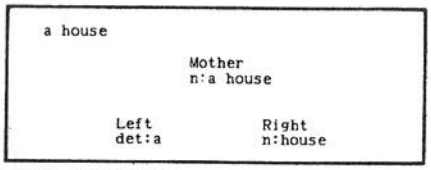


図7-2　文法規則 2

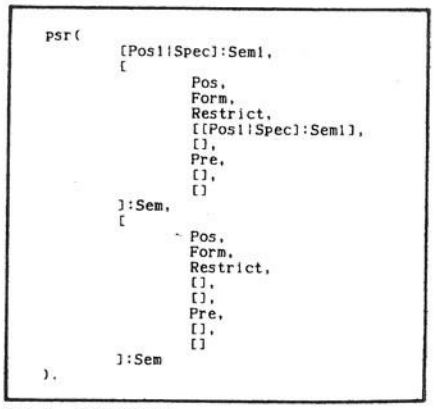


図8-1　文法規則 3

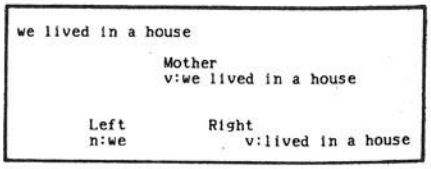


図8-2　文法規則 3

第4要素から第8要素までが，接続に関する制約を記述したものである。

文法規則の主要な役割は，この接続に関する制約を満たしているかどうかのチェックである。リストの後ろに付いている"Sem"は意味情報で，文法チェックと同時に文の意味構造を形成していくようになっている。

図7－2は，文法規則2を図で表わしたものである。文法規則2は文の前の要素が後ろの要素を修飾する関係を表わしている。"a house"の例の場合，文法カテゴリーが冠詞で

ある“a”が，名詞の“house”に付いて新たな名詞カテゴリーを生成することを表わしている。

図8－2は，文法規則3を図で表わしたものである。文法規則3は文の後ろの要素が前の要素を支配する関係を表わしている。“We lived in a house”の例の場合，文法カテゴリーが動詞である“lived in a house”が前方に名詞の“we”を要求して，新たな動詞カテゴリーを生成することを表わしている。

図9－2は，文法規則4を図で表わしたものである。文法規則4は，文の前の要素が後ろの要素を支配する関係を表わしている。“in a house”の例の場合，文法カテゴリーが前置詞である“in”が，後方に名詞の“a house”を要求して，新たな前置詞カテゴリーを生成することを表わしている。

そのほかに，省略や関係詞を記述するのに必要な非制限依存に関する文法ルールもあるが，パソコン上のシステムでは実行スピードの点から問題があるために割愛した。

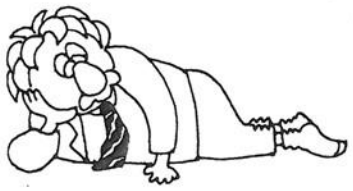
実行スピードの点であるが，上記の文法記述では少し複雑な文になると，もう戻って来ないのではと思うくらいに長い時間がかかってしまう。そこで，実際のインプリメントでは，文法規則の品詞記述の部分になにが入り得るかを制約として記述することで，たいていの文章ならば数秒以内で結果が返るようにした。

今回の最長時間は，VXの286モード上のProlog－KABAで15秒である。

図10が，構文解析システムの全プログラムである。

今回採用したパーサーは，ボトムアップ・パーサーを採用している。解析スピードを上げるために，トップダウン予測というテクニックを用いている。ボトムアップ・パーサーについては，前回の解説を参照していただきたい。

また，コメントとして接続詞に関する文法カテゴリーが記述してあるが，このコメントの部分をはずして前の述語とくっ付けると接続詞も扱えるようになる。このようなことをしたのも，計算量を減らすためである。



```
psr(
        [
          Pos,
                Form,
                Restrict,
                Spec,
                [[Pos|Subcat]:Sem|Tail],
                Post,
                Pre,
                []
        ]:Sem,
        [Pos|Subcat]:Sem,
        [
          Pos,
          Form,
          Restrict,
          Spec,
          Tail,
          Post,
          Pre,
          []
        ]:Sem
).
```

図9-1　文法規則 4

```
                Mother
                p:in a house


        Left            Right
        p:in            n:a house
```

図9-2　文法規則 4

```
%************************%
%       オペレータ定義   %
%************************%
:-op(100,xfy,':').
%************************%
%       パーサー         %
%************************%
hpsg(Top,C1,Cn,S1,Sn) :-
        lex(Lex,C1,C2,S1,S2),
        connect(Lex,Top,C2,Cn,S2,Sn).
connect(Top,Top,X,X,Y,Y).

connect(Left,Top,C1,Cn,S1,Sn) :-
        psr(Left,Right,Mother),
        link(Mother,Top),
        hpsg(Right,C1,C2,S1,S2),
        connect(Mother,Top,C2,Cn,S2,Sn).
%*********************************%
%       トップダウン予測          %
%*********************************%
link([X|_]:_,[X|_]:_).
link([det|_]:_,[n|_]:_).
link([a|_]:_,[n|_]:_).
```

図10　構文解析プログラム

（次ページに続く）

```
link([nl_]:_,[vl_]:_).
link([detl_]:_,[vl_]:_).
link([al_]:_,[vl_]:_).
link([pl_]:_,[vl_]:_).
link([nl_]:_,[juncl_]:_).
link([vl_]:_,[juncl_]:_).
link([al_]:_,[juncl_]:_).
/*
link([juncl_]:_,[vl_]:_).
*/
%************************%
%       辞書引き         %
%************************%
lex(Lex,C1,C2,[Word|T],T) :-
        M =.. [Word,Lex,C1,C2],
        M.
%************************%
%       文法規則         %
%************************%
psr(
        [Pos1|Spec]:Sem1,
        [Pos,_,_,[],[],[[Pos1|Spec]:Sem1],[],[]]:Sem,
        [Pos1|Spec]:Sem
) :-
        [Pos1,Pos] = [n,n];
        [Pos1,Pos] = [n,a];
        [Pos1,Pos] = [n,p];
        [Pos1,Pos] = [v,p];
        [Pos1,Pos] = [v,a].
/*
;
        [Pos1,Pos] = [n,junc];
        [Pos1,Pos] = [v,junc];
        [Pos1,Pos] = [a,junc].
*/

psr(
        [Pos,_,_,[],[],[],[[Pos1|Spec]:Sem1],[]]:Sem,
        [Pos1|Spec]:Sem1,
        [Pos1|Spec]:Sem
) :-
        [Pos,Pos1] = [det,n];
        [Pos,Pos1] = [p,v];
        [Pos,Pos1] = [a,n];
        [Pos,Pos1] = [a,v].
psr(
        [Pos1|Spec]:Sem1,
        [Pos,Form,Restrict,[[Pos1|Spec]:Sem1],[],Pre,[],[]]:Sem,
        [Pos,Form,Restrict,[],[],Pre,[],[]]:Sem
) :-
        [Pos1,Pos] = [n,v];
        [Pos1,Pos] = [n,a].

psr(
        [Pos,Form,Restrict,Spec,[[Pos1|Subcat]:Sem1|Tail],Post,Pre,[
]]:Sem,
        [Pos1|Subcat]:
[]]:Sem   [Pos,Form,Restrict,Spec,Tail,Post,Pre,
) :-
        [Pos,Pos1] = [v,n];
        [Pos,Pos1] = [v,v];
        [Pos,Pos1] = [v,p];
        [Pos,Pos1] = [v,a];
        [Pos,Pos1] = [p,n];
        [Pos,Pos1] = [a,n].
/*
;
        [Pos,Pos1] = [junc,n];
        [Pos,Pos1] = [junc,v];
        [Pos,Pos1] = [junc,a].
*/
%********************************%
%       ユーティリティー         %
%********************************%
plural(X) :-
        member(X,[there,you,we,they,p1,p2,p3]).
singular(X) :-
        member(X,[there,it,i,you,he,she,s1,s2,s3]).
member(A,[A|_]).
member(A,[_|X]) :-
        member(A,X).
%********************************%
%       英文入力パッケージ       %
%********************************%
input(Ret) :-
      ascread(X),
        word_cut(X,Y),
        naming(Y,Ret),
        !.
naming([],[]).
naming([X|Y],[Z|W]) :-
        name(Z,X),
        naming(Y,W).
%************************%
%       単語切り出し       %
%************************%
word_cut([],[]).
word_cut([32|X],Y) :-
    word_cut(X,Y).
word_cut([44|X],[[44]|Y]) :-
        word_cut(X,Y).
word_cut([46|X],[[46]|Y]) :-
        word_cut(X,Y).
word_cut([33|X],[[33]|Y]) :-
        word_cut(X,Y).
word_cut([63|X],[[63]|Y]) :-
        word_cut(X,Y).
word_cut(X,[Word|Y]) :-
        word_cut1(X,Word,Tail),
        word_cut(Tail,Y).
word_cut1([],[],[]).
```

図10　構文解析プログラム（続き）

```
word_cut1([32|Tail],[],[32|Tail]) :-
        !.
word_cut1([33|Tail],[],[33|Tail]) :-
        !.
word_cut1([44|Tail],[],[44|Tail]) :-
        !.
word_cut1([46|Tail],[],[46|Tail]) :-
        !.
word_cut1([Code|X],[Code|Word_Rest],Tail) :-
        word_cut1(X,Word_Rest,Tail).
%************************%
%        名詞辞書        %
%************************%
'Tom'(
        [
                n,
                s3,
                -,
                [],
                [],
                [],
                [],
                []
        ]
        :
        トム,
        N,N
).
'Kim'(
        [
                n,
                s3,
                -,
                [],
                [],
                [],
                [],
                []
        ]
        :
        キム,
        N,N
).
'This'(
        [
                n,
                s3,
                nom,
                [],
                [],
                [],
                [],
                []
        ]
        :
        これ,
        N,N
).
'summer'(
        [
                n,
                s3,
                -,
                [],
                [],
                [],
                [],
                []
        ]
        :
        夏,
        N,N
).
'year'(
        [
                n,
                s3,
                -,
                [],
                [],
                [],
                [],
                []
        ]
        :
        年,
        N,N
).
        [
                n,
                pl,
                nom,
                [],
                [],
                [],
                [],
                []
        ]
        :
        我々,
        N,N
).
'house'(
        [
                n,
                s3,
                -,
                [],
                [],
                [],
                [],
```

図10　構文解析プログラム（続き）

（次ページに続く）

```
                [],
                [],
                [],
                [[n,s3,_,[],[],[],[],[]]:Sem],
                []
        ]
        :
        Sem,
        N,N
).
%************************%
%       形容詞・副詞       %
%************************%
'silly'(
        [
            a,
                fin,
                -,
                [[n,s3,nom,[],[],[],[],[]]:X],
                [],
                [],
                [],
                []
        ]:
        ばか(X),
        N,N
).
'late'(
        [
                a,
                fin,
                -,
                [],
                [],
                [],
                [[n,s3,Case,[],[],[],[],[]]:X],
                []
        ]
        :
        [遅い,X],
        N,N
).
'that'(
        [
                n,
                PN,
                -,
                [],
                [[v,_,_,_,_,[],[],[],[]]:Y],
                [[n,PN,_,[],[],[],[],[]]:X],
                [],
                []
        ]
        :
        [Y,X],
        N,N
).
'that'(
        [
                a,
                fin,
                -,
                [],
                [],
                [],
                [[n,s3,Case,[],[],[],[],[]]:X],
                    []
    ]
        :
        [その,X],
        N,N
).
%****************%
%       前置詞       %
%****************%
in(
       [
          p,
                -,
                in,
                [],
                [[n,s3,acc,[],[],[],[],[]]:Y],
                [[n,_,_,[],[],[],[],[]]:X],
                [],
                []
        ]
        :
        [Y,の中の,X],
        N,N
).
in(
        [
                p,
                -,
                in,
                [],
                [[n,s3,acc,[],[],[],[],[]]:Y],
                [[v,Form,Restrict,[],[],[],[],[]]:X],
                [],
                []
        ]
        :
        [Y,に,X],
        N,N
).
in(
        [
                p,
                -,
                in,
                [],
                [[n,s3,acc,[],[],[],[],[]]:Y],
```

図10　構文解析プログラム（続き）

```
                [],
                [[v,Form,_,[],[],[],[],[]]:X],
                []
        ]
        :
        [Y,に,X],
        N,N
).
to(
        [
                p,
                to,
                _,
                [],
                [[n,_,acc,[],[],[],[],[]]:Y],
                [],
                [],
                []
        ]
        :
        Y,
        N,N
).
of(
        [
                p,
                _,
                of,
                [],
                [[n,s3,acc,[],[],[],[],[]]:Y],
                [[n,s3,_,[],[],[],[],[]]:X],
                [],
                []
        ]
        :
        [Y,の,X],
        N,N
).
%################%
%       接続詞        %
%################%
'and'(
      [
         junc,
                _,
                and,
                [],
                [[n,Form,Case,[],[],[],[],[]]:Y],
                [[n,Form,Case,[],[],[],[],[]]:X],
                [],
                []
        ]
        :
        [X,と,Y],
        N,N
).
```

```
%########################%
%       テストルーチン        %
%########################%
ex :-
       input(X),
  hpsg(Sem,C,[],X,[]),
        write(C),nl,
        write(Sem),nl,nl,
        fail.
ex.
ex1 :-
        hpsg(Sem,C,[],[in,the,late,summer,of,that,year,we,lived,in,a
,house],[]),
        write(C),nl,
        write(Sem),nl,nl,
        fail.
ex1.
ex2 :-
        hpsg(Sem,C,[],['Tom',loves,'Kim'],[]),
        write(C),nl,
        write(Sem),nl,nl,
        fail.
ex2.
ex3 :-
        hpsg(Sem,C,[],['This',is,a,pen],[]),
        write(C),nl,
        write(Sem),nl,nl,
        fail.
ex3.
ex4 :-
        hpsg(Sem,C,[],['Tom',is,silly],[]),
        write(C),nl,
        write(Sem),nl,nl,
        fail.
ex4.
ex5 :-
        hpsg(Sem,C,[],['Tom',tries,to,walk],[]),
        write(C),nl,
        write(Sem),nl,nl,
        fail.
ex5.
ex6 :-
        hpsg(Sem,C,[],['Tom',seems,to,walk],[]),
        write(C),nl,
        write(Sem),nl,nl,
        fail.
ex6.
ex7 :-
        hpsg(Sem,C,[],['Tom',promises,'Kim',to,walk],[]),
        write(C),nl,
        write(Sem),nl,nl,
        fail.
ex7.
ex8 :-
        hpsg(Sem,C,[],['Tom',seems,to,love,'Kim'],[]),
```

図10　構文解析プログラム（続き）

（次ページに続く）

```
        ]
        :
        家,
        N,N
).
pen(
        [
                n,
                s3,
                _,
                [],
                [],
                [],
                [],
                []
        ]
        :
        ペン,
        N,N
).
%################%
%       動詞       %
%################%
is(
      [
          v,
              fin,
              _,
              [[n,s3,nom,[],[],[],[],[]]:X],
              [[n,s3,nom,[],[],[],[],[]]:Y],
              [],
              [],
              []
        ]
        :
            isa(X,Y),
        N,N
).
is(
   [
              v,
              fin,
              _,
              [[n,s3,nom,[],[],[],[],[]]:X],
              [[a,_,_,[[n,s3,_,[],[],[],[],[]]:X],[],[],[],[]]:Y],
              [],
              [],
              []
        ]
        :
        Y,
        N,N
).
walk(
        [
              v,
              base,
              _,
              [[n,_,_,[],[],[],[],[]]:X],
              [],
              [],
              [],
              []
        ]
        :
        歩く(agent:X),
        N,N
).
love(
        [
              v,
              base,
              _,
              [[n,s3,nom,[],[],[],[],[]]:X],
              [[n,_,acc,[],[],[],[],[]]:Y],
              [],
              [],
              []
        ]
        :
        愛する(agent:X,patient:Y),
        N,N
).
loves(
        [
              v,
              fin,
              _,
              [[n,s3,nom,[],[],[],[],[]]:X],
              [[n,_,acc,[],[],[],[],[]]:Y],
              [],
              [],
              []
        ]
        :
        愛する(agent:X,patient:Y),
        N,N
).
tries(
        [
              v,
              fin,
              _,
              [[n,s3,nom,[],[],[],[],[]]:X],
              [[v,to,_,[[n,s3,_,[],[],[],[],[]]
,                                  ]:X],[],[],[],[]]:Y]
              [],
              [],
              []
        ]
        :
```

図10　構文解析プログラム（続き）

```
        試みる(agent:X,goal:Y),
        N,N
).
seems(
        [
                v,
                fin,
                _,
                [[n,s3,nom,[],[],[],[],[]]:X],
                [[v,to,_,[[n,s3,_,[],[],[],[],[]]:X],[],[],[],[]]:Y]

                [],
                [],
                []
        ]
        :
        思える(situation:Y),
        N,N
).
promises(
        [
                v,
            fin,
                _,
                [[n,s3,nom,[],[],[],[],[]]:X],
                [
                        [n,_,acc,[],[],[],[],[]]:Y,
                        [v,to,_,[[n,s3,_,[],[],[],[],[]]:X],[],[],[]
,[]]:Z

                ],
                [],
                [],
                []
        ]
        :
        約束する(agent:X,patient:Y,goal:Z),
        N,N
).
seems(
        [
                v,
                fin,
                _,
                [[n,s3,nom,[],[],[],[],[]]:X],
                [
                        [p,to,_,[],[],[],[],[]]:Y,
                        [v,to,_,[[n,s3,_,[],[],[],[],[]]:X],[],[],[]
,[]]:Z

                ],
                [],
                [],
                []
        ]
        :
        思える(situation:Z,patient:Y),
        N,N
).
to(
        [
                v,
                to,
                _,
                Spec,
                [[v,base,_,Spec,_,_,_,_]:Y],
                [],
                [],
                []
        ]
        :
        Y,
        N,N
).
'lived'(
        [
                v,
                pp,
                _,
                [[n,PN,nom,[],[],[],[],[]]:X],
                [],
                [],
                [],
                []
        ]
        :
        [X,は,住む],
        N,N
).
%################%
%        冠詞        %
%################%
'the'(
      [
          det,
              _,
              the,
              [],
              [],
              [],
              [[n,s3,Case,[],[],[],[],[]]:Sem],
              []
        ]
        :
        Sem,
        N,N
).
'a'(
        [
                det,
                a,
                _,
```

図10　構文解析プログラム（続き）

（次ページに続く）

```
        write(C),nl,
        write(Sem),nl,nl,
        fail.
ex8.
ex9 :-
        hpsg(Sem,C,[],['Tom',seems,to,'Kim',to,walk],[]),
        write(C),nl,
        write(Sem),nl,nl,
        fail.
ex9.
```

図10　構文解析プログラム（続き）

実行例

今回の辞書項目は，図11に挙げた通りである。

実行例は，図12に挙げた文章について試みることができる。それ以外にも，図11の辞書に記述した単語で構成される文章であれば，

?－ex.

と入力することで，いろいろと試すことが可能である。

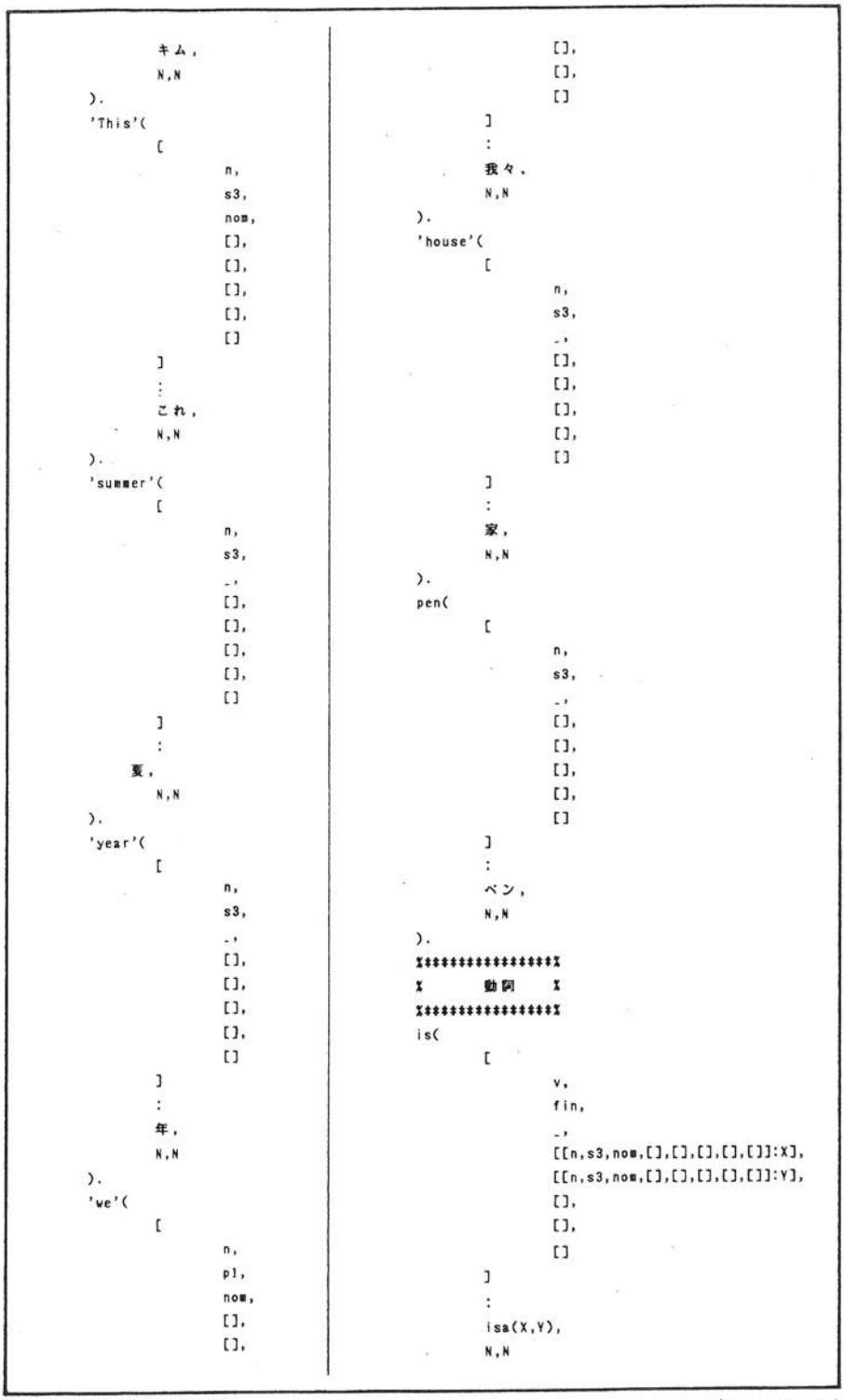
また，解析に関しては，バックトラックで得られるすべての文章を生成できるようにしてある。

図13に，図12の“ex5”から“ex9”までの実行結果をサンプルとして挙げる。

“tries”や“seems”といった，扱いのむずかしい単語についても的確な意味表現を生成している点に注目していただきたい。

```
%********************************%
%       英文入力パッケージ       %
%********************************%
input(Ret) :-
        ascread(X),
        word_cut(X,Y),
        naming(Y,Ret),
        !.
naming([],[]).
naming([X|Y],[Z|W]) :-
        name(Z,X),
        naming(Y,W).
%************************%
%       単語切り出し     %
%************************%
word_cut([],[]).
word_cut([32|X],Y) :-
        word_cut(X,Y).
word_cut([44|X],[[44]|Y]) :-
        word_cut(X,Y).
word_cut([46|X],[[46]|Y]) :-
        word_cut(X,Y).
word_cut([33|X],[[33]|Y]) :-
        word_cut(X,Y).
word_cut([63|X],[[63]|Y]) :-
        word_cut(X,Y).
word_cut(X,[Word|Y]) :-
        word_cut1(X,Word,Tail),
        word_cut(Tail,Y).
word_cut1([],[],[]).
word_cut1([32|Tail],[],[32|Tail]) :-
        !.
word_cut1([33|Tail],[],[33|Tail]) :-
        !.
word_cut1([44|Tail],[],[44|Tail]) :-
        !.
word_cut1([46|Tail],[],[46|Tail]) :-
        !.
word_cut1([Code|X],[Code|Word_Rest],Tail) :-
        word_cut1(X,Word_Rest,Tail).
%************************%
%       名詞辞書         %
%************************%
'Tom'(
        [
                n,
                s3,
                -,
                [],
                [],
                [],
                [],
                []
        ]
        :
        トム,
        N,N
).
'Kim'(
        [
                n,
                s3,
                -,
                [],
                [],
                [],
                [],
                []
        ]
        :
```

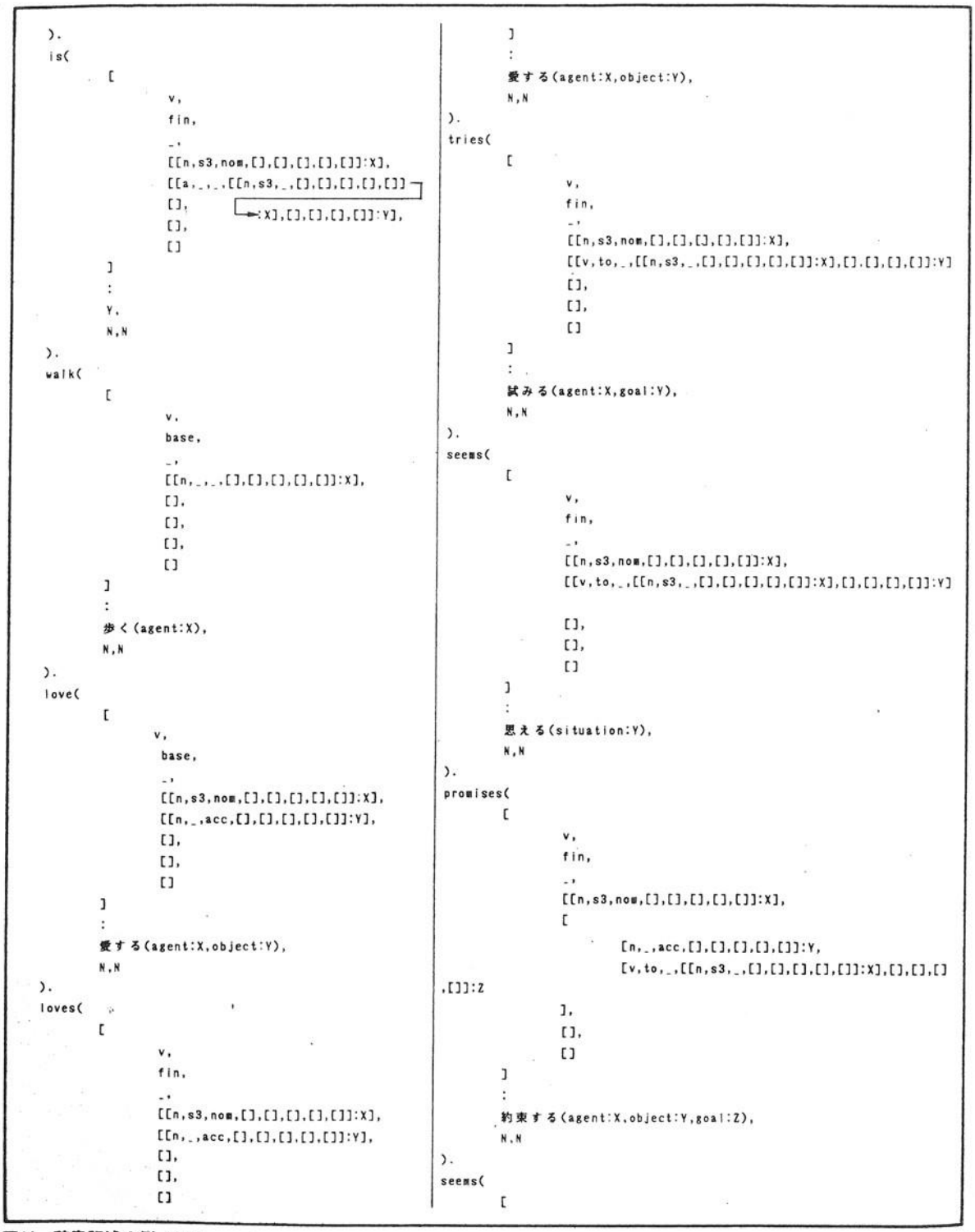
図11　辞書記述の例



```
        キム，
        N,N
    ).
    'This'(
        [
            n,
            s3,
            nom,
            [],
            [],
            [],
            [],
            []
        ]
        :
        これ，
        N,N
    ).
    'summer'(
        [
            n,
            s3,
            -,
            [],
            [],
            [],
            [],
            []
        ]
        :
      夏，
        N,N
    ).
    'year'(
        [
            n,
            s3,
            -,
            [],
            [],
            [],
            [],
            []
        ]
        :
        年，
        N,N
    ).
    'we'(
        [
            n,
            pl,
            nom,
            [],
            [],
            [],
            []
        ]
        :
        我々，
        N,N
    ).
    'house'(
        [
            n,
            s3,
            -,
            [],
            [],
            [],
            [],
            []
        ]
        :
        家，
        N,N
    ).
    pen(
        [
            n,
            s3,
            -,
            [],
            [],
            [],
            [],
            []
        ]
        :
        ペン，
        N,N
    ).
    %%%%%%%%%%%%%%%%%
    %       動詞    %
    %%%%%%%%%%%%%%%%%
    is(
        [
            v,
            fin,
            -,
            [[n,s3,nom,[],[],[],[],[],[]]:X],
            [[n,s3,nom,[],[],[],[],[],[]]:Y],
            [],
            [],
            []
        ]
        :
        isa(X,Y),
        N,N
```

図11　辞書記述の例（続き）

（次ページに続く）



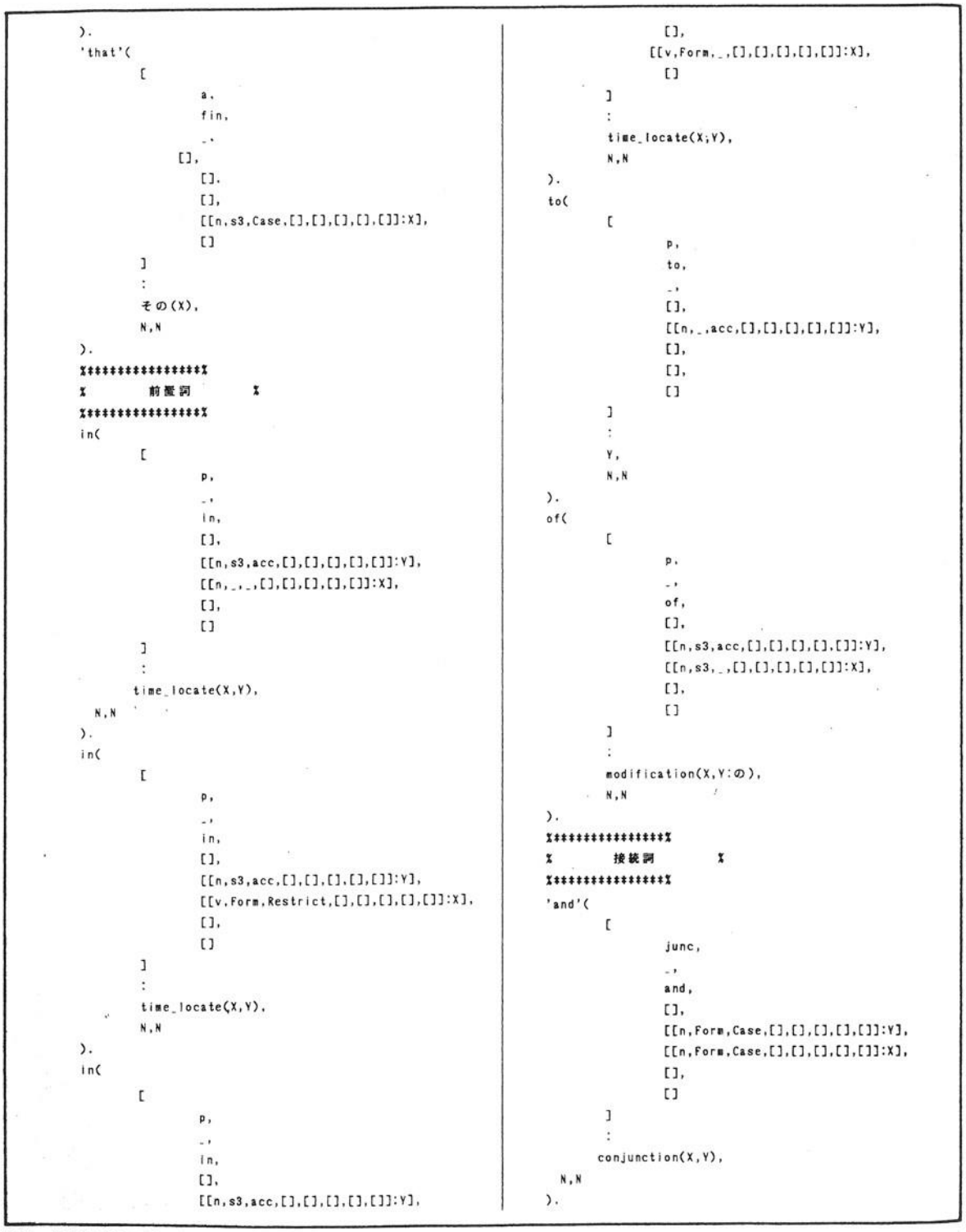
```
).
is(
        [
                v,
                fin,
                _,
                [[n,s3,nom,[],[],[],[],[]]:X],
                [[a,_,_,[[n,s3,_,[],[],[],[],[]]
                [],                 :X],[],[],[],[]]:Y],
                [],
                []
        ]
        :
        Y,
        N,N
).
walk(
        [
                v,
                base,
                _,
                [[n,_,_,[],[],[],[],[]]:X],
                [],
                [],
                [],
                []
        ]
        :
        歩く(agent:X),
        N,N
).
love(
        [
               v,
                base,
                _,
                [[n,s3,nom,[],[],[],[],[]]:X],
                [[n,_,acc,[],[],[],[],[]]:Y],
                [],
                [],
                []
        ]
        :
        愛する(agent:X,object:Y),
        N,N
).
loves(
        [
                v,
                fin,
                _,
                [[n,s3,nom,[],[],[],[],[]]:X],
                [[n,_,acc,[],[],[],[],[]]:Y],
                [],
                [],
                []
        ]
        :
        愛する(agent:X,object:Y),
        N,N
).
tries(
        [
                v,
                fin,
                _,
                [[n,s3,nom,[],[],[],[],[]]:X],
                [[v,to,_,[[n,s3,_,[],[],[],[],[]]:X],[],[],[],[]]:Y]
                [],
                [],
                []
        ]
        :
        試みる(agent:X,goal:Y),
        N,N
).
seems(
        [
                v,
                fin,
                _,
                [[n,s3,nom,[],[],[],[],[]]:X],
                [[v,to,_,[[n,s3,_,[],[],[],[],[]]:X],[],[],[],[]]:Y]

                [],
                [],
                []
        ]
        :
        思える(situation:Y),
        N,N
).
promises(
        [
                v,
                fin,
                _,
                [[n,s3,nom,[],[],[],[],[]]:X],
                [
                        [n,_,acc,[],[],[],[],[]]:Y,
                        [v,to,_,[[n,s3,_,[],[],[],[],[]]:X],[],[],[]
,[]]:Z
                ],
                [],
                []
        ]
        :
        約束する(agent:X,object:Y,goal:Z),
        N,N
).
seems(
        [
```

図11　辞書記述の例（続き）

```
                v,
                fin,
                _,
                [[n,s3,nom,[],[],[],[],[]]:X],
                [
                        [p,to,_,[],[],[],[],[]]:Y,
                        [v,to,_,[[n,s3,_,[],[],[],[],[]]:X],[],[],[]
,[]]:Z
                ],
                [],
                [],
                []
        ]
        :
        思える(situation:Z,object:Y),
        N,N
).
to(
        [
                v,
                to,
                _,
                Spec,
                [[v,base,_,Spec,_,_,_,_]:Y],
                [],
                [],
                []
        ]
        :
        Y,
        N,N
).
'lived'(
        [
                v,
                pp,
                _,
                [[n,PN,nom,[],[],[],[],[]]:X],
                [],
                [],
                [],
                []
        ]
        :
        住む(agent:X),
        N,N
).
%################%
%       冠詞    %
%################%
'the'(
        [
                det,
                _,
                the,
                [],
                [],
                [],
                [[n,s3,Case,[],[],[],[],[]]:Sem],
                []
        ]
        :
        Sem,
        N,N
).
'a'(
        [
                det,
                a,
                _,
                [],
                [],
                [],
                [[n,s3,_,[],[],[],[],[]]:Sem],
                []
        ]
        :
        Sem,
        N,N
).
%########################%
%       形容詞・副詞    %
%########################%
'silly'(
        [
                a,
                fin,
                _,
                [[n,s3,nom,[],[],[],[],[]]:X],
                [],
                [],
                [],
                []
        ]:
        ばか(X),
        N,N
).
'late'(
        [
                a,
                fin,
                _,
                [],
                [],
                [],
                [[n,s3,Case,[],[],[],[],[]]:X],
                []
        ]
        :
        遅い(X),
        N,N
```

図11　辞書記述の例（続き）

（次ページに続く）



```
).
'that'(
        [
                a,
                fin,
                _,
             [],
                [],
                [],
                [[n,s3,Case,[],[],[],[],[]]:X],
                []
        ]
        :
        その(X),
        N,N
).
%################%
%        前置詞        %
%################%
in(
        [
                p,
                _,
                in,
                [],
                [[n,s3,acc,[],[],[],[],[]]:Y],
                [[n,_,_,[],[],[],[],[]]:X],
                [],
                []
        ]
        :
       time_locate(X,Y),
  N,N
).
in(
        [
                p,
                _,
                in,
                [],
                [[n,s3,acc,[],[],[],[],[]]:Y],
                [[v,Form,Restrict,[],[],[],[],[]]:X],
                [],
                []
        ]
        :
        time_locate(X,Y),
        N,N
).
in(
        [
                p,
                _,
                in,
                [],
                [[n,s3,acc,[],[],[],[],[]]:Y],
                [],
                [[v,Form,_,[],[],[],[],[]]:X],
                []
        ]
        :
        time_locate(X,Y),
        N,N
).
to(
        [
                p,
                to,
                _,
                [],
                [[n,_,acc,[],[],[],[],[]]:Y],
                [],
                [],
                []
        ]
        :
        Y,
        N,N
).
of(
        [
                p,
                _,
                of,
                [],
                [[n,s3,acc,[],[],[],[],[]]:Y],
                [[n,s3,_,[],[],[],[],[]]:X],
                [],
                []
        ]
        :
        modification(X,Y:の),
        N,N
).
%################%
%        接続詞        %
%################%
'and'(
        [
                junc,
                _,
                and,
                [],
                [[n,Form,Case,[],[],[],[],[]]:Y],
                [[n,Form,Case,[],[],[],[],[]]:X],
                [],
                []
        ]
        :
       conjunction(X,Y),
  N,N
).
```

図11　辞書記述の例（続き）