拡張LINGOLのn進木への拡張

畝見 達夫* 田中 穂積** 市川 惇信*
*東京工業大学システム科学専攻 **電子技術総合研究所

まえがき

コンピュータによる自然言語処理は様々な面で、その重要性を増しつつあるが、本研究では、そのための道具の一つとして、1978年に電総研で開発された「拡張LINGOL」[3]をベースに、より柔軟な文法規則表現が可能なパーザを基礎とする自然言語処理のためのプログラミングシステムを作成した。

LINGOL[1,2]は文脈自由文法を基礎にしてはいるものの、実際に計算機上で動かすという都合上、各文法規則における右側非終端記号の記述個数を高々2つに制限しており、そのため、非終端記号及び文法規則の数が増し、文法大系の記述が繁雑になるという欠点を有していた。本研究では、その記述個数制限をなくし、それに加えて、非終端記号の不定数回繰り返しの指定も許すこととし、より柔軟な文法表現を可能にした。本システムによる構文解析木がn進木となることから、これを「n進木LINGOL」と呼ぶことにする。尚、こういった機能拡張に伴い、システム自体のプログラムをほぼ全面的に作成し直す結果となった。

1 n進木LINGOLへの拡張

LINGOL及び拡張LINGOLでは、各文法規則における右側非終端記号の記述個数が高々2つまでという制限のために、文法大系の記述が繁雑になるという欠点があった。例えば次のような文法規則があるとしよう。

$$A \rightarrow B\ C\ \text{と}\ D \cdots \text{す}\ E\ \text{と}\ F \cdots \text{す} \qquad \cdots\cdots\cdots\cdots (1)$$

これをLINGOLにのる形の文法規則とするためには、例えば次のように書かなければならない。

$$\left.\begin{array}{llll} A \rightarrow B\,C_1 & C_2 \rightarrow C\,D_1 & E_1 \rightarrow E & F_1 \rightarrow F\,F_1 \\ C_1 \rightarrow C_2\,E & D_1 \rightarrow D & E_1 \rightarrow E\,F_1 & \\ C_2 \rightarrow C & D_1 \rightarrow D\,D_1 & F_1 \rightarrow F & \end{array}\right\} (2)$$

(1)の形式の文法は次の例に見られるように、実際の構文解析においてしばしば現われる。

① 日本語の係り結び ex. まるで…ような
② 数式の2項演算子 <項> → <項> <2項演算子> <項> ;
③ カッコ <項> → <左カッコ> <式> <右カッコ> ;
④ 日本語における修飾語 <文> → と <副詞句> …す <動詞> ;
<名詞句> → と <形容詞句> …す <名詞> ;

①～③は、文法規則の右側非終端記号を任意個にすることを、④は、不定数回繰り返しの指定を可能にすることを要請する。

このことから、拡張LINGOLに対し、これらの機能を追加拡張することとし、それに伴なって、文法規則の具体的な記述形式、構文解析木の内部表現形式、拡張LINGOLの特徴である予測制御のためのadvice部及び、構文解析の結果としてできるプログラムを構成するsem部に使用するための関数に対し、大幅な変更を加えた。新しい拡張機能の活用により次の効果が期待できる。

① 非終端記号の数が削減される。

② 構文規則の数が削減される。

③ 構文解析木の構造を、文章の意味構造に近づけることが容易になる。

この機能拡張は、文法記述能力に対して何ら変化をもたらすものではなく、単にプログラミングを容易にするだけであるが、自然言語の文法を作り上げる上でこのことは実用的な意味で、極めて重要であろう。

2 基本動作

では本システムπ進木LINGOLの実際の動作について説明しよう。全体の基本的動作の流れは拡張LINGOLと同様右図（図2-1）のようになっている。以下、各処理について述べる。

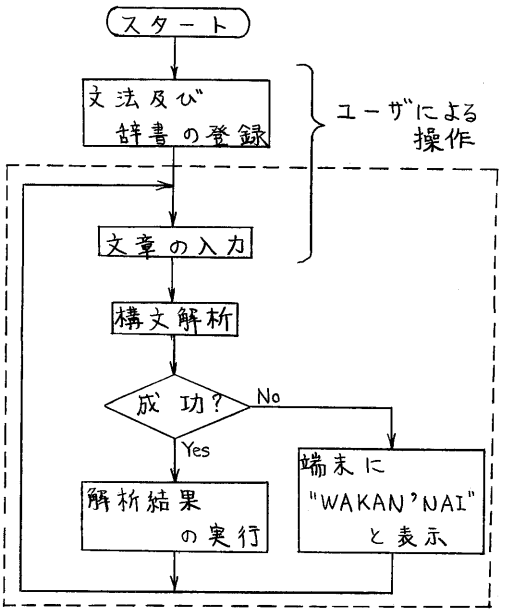


図.2-1 π進木LINGOL
の全体的な流れ図

2.1 文法及び辞書の登録

LINGOLでは構文規則を終端記号を一つの非終端記号に結び付ける辞書項目と非終端記号のみからなる文法規則とに分けている。具体的には各々の規則は下のようなS-式で記述される。

辞書項目

（a A（<message> <cog>） <sem>）

a：終端記号（単語）。

A：非終端記号（文法カテゴリー名）。

<message>：次に述べる文法規則中のadvice部によって参照され、予測制御による例外処理に利用されるデータのリスト。

<cog>：あいまいさを処理するための数値。

<sem>：構文解析の結果としてできるプログラムの構成要素となるS-式。

文法規則

（N_0（{*}（N_1 <adv_1>）
{*}（N_2 <adv_2>）
⋮ ⋮
{*}（N_n <adv_n>））<cog> <sem>）

N_0：この文規則によってできる部分構文解析木（PPT）の根に当るノードにくる非終端記号（左側非終端記号）。

$N_1 \sim N_n$ ：右側非終端記号。

＊：そのすぐ右側の非終端記号に対して不定数回繰り返しを指定する印。

<advi>：構文解析実行時に、これと対になっている右側非終端記号の下に完成済の別のPPTを連結する時点で評価（EVAL）される手続き。その評価結果が真（非NIL）の場合は連結しない。

advice部及びmessageの使用方法については拡張LINGOLに関する文献3)，5)に詳しい。

以上のような書式で記述された各構文規則は、文法規則の場合は、最も左側の右側非終端記号、辞書項目の場合は、その終端記号をP-NAMEとするアトムの属性リスト中に記録される。但し、文法規則において、左端の右側非終端記号に対し不定数回繰り返しの指定があるときは、次のように展開する。

① （N_0 （＊（N_1 <adv1>）（N_2 <adv2>）・・・） <cog> <sem>）

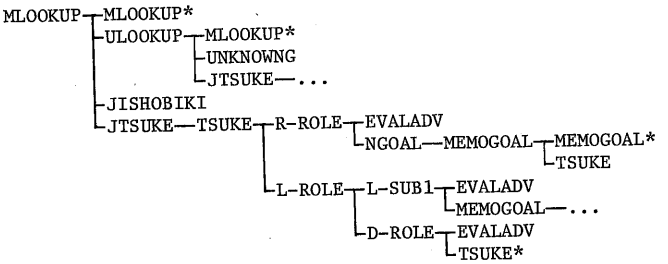
$\Rightarrow$ { （N_0 （（N_1 <adv1>）＊（N_1 <adv1>）（N_2 <adv2>）・・・）<cog><sem>）
　　（N_0 （（N_2 <adv2>）・・・） <cog> <sem>）

② （N_0 （＊（N_1 <adv1>））） <cog> <sem>）

$\Rightarrow$ （N_0 （（N_1 <adv1>）＊（N_1 <adv1>））） <cog> <sem>）

2.2 入力文の構文解析

解析すべき言語の構文規則を登録した後、入力モードを文入力モードに切り換え、解析すべき文章を入力すれば、その構文規則に従って構文解析が行なわれる。構文解析を行なう関数は下図（図2-2）のように互いに再帰的に呼び合いながら構文解析木を左枝側から bottom-up と top-down を繰り返しつつ組み上げる。



＊印は再帰的に呼び出されている部分。

MLOOKUP　：自動分かち書きを伴う単語（終端記号）の切り出し。
ULOOKUP　：未定義語の切り出し。
UNKNOWNG　：未定義語処理のための一時的な辞書項目の作成。
JISHOBIKI：辞書の検索。
JTSUKE　：辞書項目のPPT化。
TSUKE　：完成済PPTの連結。
R-ROLE　：未解決な枝（GOAL）へのPPTの連結。
L-ROLE　：PPTを左端の枝（右側非終端記号）に連結可能な文法規則の適用。
L-SUB1　：〃　〃　〃　〃連結した未解決PPTの作成。

D-ROLE : 右側非終端記号が１つしかない文法規則の適用。
NGOAL : 部分的に解決したGOALsの作成。
MEMOGOAL : 新しいGOALの登録。
EVALADV : advice部の評価。

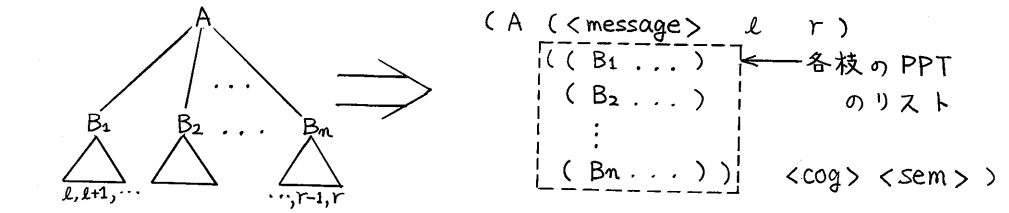
図2-2　構文解析を行なう関数とその参照関係

2.2.1　単語の切り出し

入力された文字列から単語を切り出す作業については、自動分かち書き、未定義語の抽出、数字の処理等、拡張LINGOLが持っている機能をほぼ踏襲している。

2.2.2　PPTの内部表現

完成済のPPTはマシン上で次のように表現される（図2-3）。



A：左側非終端記号　　l：PPTの左端の単語番号
Bi：右側非終端記号　　r：PPTの右端の単語番号

図2-3　完成済PPTの内部表現形式

単語番号は切り出された１つ１つの単語に対して、入力文の先頭から順に、1, 2, 3, … というふうに付けられる。

引用された辞書項目も１つのPPTと見なし、次のように変形される。

辞書項目（α　A（<message> <cog>） <sem>）
⟹ PPT（A（<message> w w） α <cog> <sem>）
α：終端記号
A：非終端記号
w：αの単語番号

2.2.3　文法規則の引用と予測文法カテゴリーの記録

１つのPPTが完成すると、関数TSUKEにより更に木を成長させるべく、そのPPTを引数として、関数R-ROLE及びL-ROLEが実行される。R-ROLEについては次の2.2.4で述べる。L-ROLEでは、完成済PPTの根に当る左側非終端記号を左端の枝とする全ての文法規則が引用され、適用が試みられる。但し、その枝に付属しているadvice部の評価結果が非NILの場合は適用しない。

適用する文法規則の右側非終端記号がただ1個のときは、関数D-ROLEにより、適用後、直ちに新たなPPTが完成され、再び、同じ処理が再帰的に繰り返される。（図2-4，左側参照）

複数の枝（右側非終端記号）を持つ場合は、関数L-SUB1により、その左端の枝を除く各枝の非終端記号を、後で下側の完成済PPTに連結さるべき予測文法カテゴリーとして記録する。（図2-4，右側参照）

但し、左端の予測文法カテゴリー（図2-4右側の図ではC）に対して不定数回繰り返しが指定されている場合は、図2-5のように展開し、2つの予測文法カテゴリーの組を作る。許容される繰り返し回数は0～N回である。

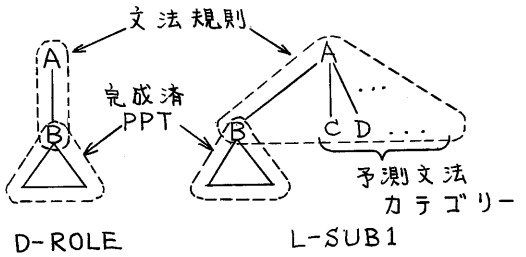


図2-4　L-ROLEにおける文法規則適用によるPPTの成長

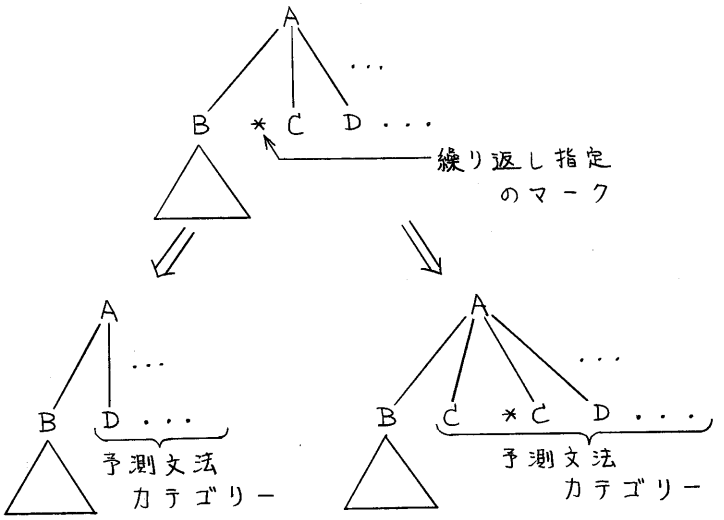


図2-5　L-SUB1における予測文法カテゴリーの不定数回繰り返し指定に基づく展開

2.2.4　予測文法カテゴリーとのマッチング

関数R-ROLEでは、完成したPPTを当てはめるべき予測文法カテゴリーを検索し、見つかればその未解決枝に対し、連結を試みる。検索のキーはPPTの根の非終端記号と左端単語番号（図2-6中のB_kとl）である。R-ROLEを実行した結果、PPTが完成すれば、

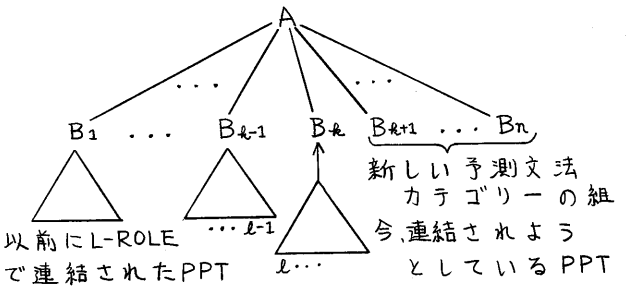


図2-6　R-ROLEにおけるPPTの連結

D-ROLEの場合と同様、再帰的に関数TSUKE を呼び出す。

また、実行後残された左端の予測文法カテゴリーに対して不定数回繰り返しの指定がなされている場合は、L-SUB1と同様、２つの予測文法カテゴリーの組に展開する。

2.2.5 advice部の利用

本システムは拡張LINGOLにそなわっている予測制御の機能を踏襲しているが、n進木への拡張に伴ない、その文法規則中での記述方法及び使用する関数を大幅に変更している。advice部は、各文法規則中の各右側非終端記号に対応して記述される。実行時では、その枝に完成済PPTが連結されようとする時点で評価され、その値が非NILの場合は連結が中止される。

advice部に書くための関数として、次のものを用意した。

① メッセージの送信

PM ：上位（根側）のノードに引数の値をメッセージとして送る。

② メッセージの削除

RM ：上位のノードにあるメッセージの中から引数で指定されたものを削除する。

③ メッセージの参照

MM ：自分自身の枝に連結されるPPTのメッセージを取り出す。

FM ：引数で指定した枝のノードにあるメッセージを取り出す。

ROOTMSG ：上位のノードにあるメッセージを取り出す。

LASTMSG ：自分自身のすぐ左側の枝のノードにあるメッセージを取り出す。

④ メッセージの検査

MSGCHK ：上位のノードにあるメッセージの中に、引数と同じものが存在するかどうか調べる。

MSGOR ：MSGCHKのOR 版。複数のメッセージ要素のうちのどれか１つでも存在するかどうか調べる。

2.3 解析結果の実行

構文解析が成功し、最終構文解析木が完成すると同時に、木の構造に従って作られたプログラムが実行（EVAL）される。プログラムは根の方の sem部がメイン、枝の方のsem部がサブとなる木構造になる。

sem部に書くための関数としては次のものを用意したが、advice部と同様、n進木への拡張に伴い、大幅に変更した。

① ＤＯ：引数で指定した枝の sem部を実行する。

② DOALL：全ての枝、または引数で指定した非終端記号を持つ全ての枝のsem部を実行し、それらの値のリストを返す。

③ MSEVAL ：引数としてPPTをとり、その sem部を実行する。

3 ユティリティ機能

自然言語処理を行なうに際し、構文規則の改変や解析プロセスの表示などを支援する諸機能を充実させることは必須であろう。n進木LINGOLでは、拡張LIN

GOLにそなわっているユティリティ機能をほぼ踏襲しており、構文規則の追加、削除、出力、PPT及び最終構文解析木の表示、CPUタイムの表示などが行なえるようになっている。

4 応用例

本システムは進木LINGOLを用いて算術式と簡単な日本語の処理を行なってみたものを次に示す。

4.1 算術式

ALGOLやPASCAL等のプログラミング言語に使われる算術式及びそれを伴なう代入文の構文解析と、計算、代入を実行するシステムを作成した。数式は、極特徴的な人工言語であり、構文解析の機能を確任するための例題としては好材料である。作成した辞書と文法及び実際の処理例を次に示す。

① 辞書

```
(:= ASGN (NIL 0) NIL)                          (** OP2 ((**) 0) (FUNCTION EXPT))
(+ OP1 (NIL 0) (FUNCTION EVAL))                (/( LK (NIL 0) NIL)
(- OP1 (NIL 0) (FUNCTION MINUS))               (/) RK (NIL 0) NIL)
(+ OP2 ((+) 0) (FUNCTION PLUS))                (/, COMMA (NIL 0) NIL)
(- OP2 ((+ -) 0) (FUNCTION DIFFERENCE))        (MOD FUNC2 (NIL 0) (FUNCTION REMAINDER))
(* OP2 ((*) 0) (FUNCTION TIMES))               (ABS FUNC1 (NIL 0) (FUNCTION ABS))
(% OP2 ((* %) 0) (FUNCTION QUOTIENT))          (/. END (NIL 0) NIL)
```

② 文法

```
(TERM ((VAR (MEMQ 'U (MM)))) 0.) (GET (DO 1.) 'ATAI))
(TERM ((NUMB NIL)) 0. (DO 1.))
(TERM ((LK NIL) (TERM NIL) (RK NIL)) 0. (DO 2.))
(TERM ((OP1 NIL) (TERM (MM))) 0. (APPLY (DO 1.) (NCONS (DO 2.))))
(TERM ((TERM NIL)
       (OP2 (OR (EQUAL (FM 1.) (MM)) (GREATERP (PRIO (FM 1.)) (PRIO (MM))) (PM (MM)) ))
       (TERM (NOT (GREATERP (PRIO (FM 2.)) (PRIO (MM)))))
     * (OPTERM (NOT (EQUAL (FM 2.) (MM)) )) )
  0.  (APPLY (DO 2.) (CONS (DO 1.) (CONS (DO 3.) (DOALL 'OPTERM)))) )
(OPTERM ((OP2 (PM (MM)))
         (TERM (NOT (GREATERP (PRIO (FM 1.) (MM)) ))) )  0. (DO 2.))
(TERM ((FUNC1 NIL) (LK NIL) (TERM NIL) (RK NIL)) 0.
      (APPLY (DO 1.) (NCONS (DO 3.))) )
(TERM ((FUNC2 NIL) (LK NIL) (TERM NIL) (COMMA NIL) (TERM NIL) (RK NIL)) 0.
      (APPLY (DO 1.) (LIST (DO 3.) (DO 5.))) )
(BUN ((TERM NIL) (END NIL)) 0.
     (PROGN (TERPRI) (PRINC "Kotae wa ") (PRINC (DO 1.)) (PRINC " desu.") (TERPRI)))
(VAR ((UNKNOWN (PM 'U))) 0.
     (PROG (X) (DICTENETR (LIST (SETQ X (MKATOM (DO 1.))) 'VAR '(NIL 0) (LIST 'QUOTE X)))
               (RETURN X)))
```

```
(BUN ((VAR NIL) (ASGN NIL) (TERM NIL) (END NIL)) 0.
     (PROG (X Y) (PUTPROP (SETQ X (DO 1.)) (SETQ Y (DO 3.)) 'ATAI)
                 (TERPRI) (PRINC X) (PRINC " ni.") (PRINC Y) (PRINC " wo Dainyuu shimashita.")
                 (TERPRI)))
```

各非終端記号は次の意味である。

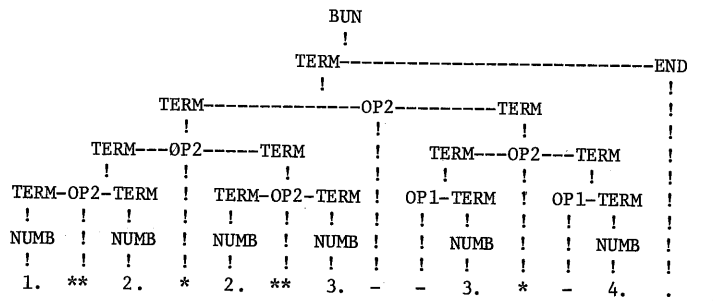
ASGN	: 代入記号	OP1	: 単項演算子
BUN	: 文（最終構文解析木の根）	OP2	: 二項演算子
COMMA	: コンマ	OPTERM	: 前に二項演算子のついた項
END	: 文の終り	RK	: 右カッコ
FUNC1	: 引数が1つの関数名	TERM	: 項あるいは式
FUNC2	: 引数が2つの関数名	UNKNOWN	: 未定義語
LK	: 左カッコ	VAR	: 変数
NUMB	: 数字（定数）		

この例題では、二項演算子の区別にmessageの情報を用い、優先順位の判定と、同じ演算子の連続の判定を行なっている。
Sem部で式の計算及び代入が行なわれる。

③ 処理例

```
? (HELLO)↵
BUN wo IRE te KUDASAI.
$ 1**2*2**3--3*-4. ↵

Parsin Time=425.ms G.C.Time=139.ms
PAGE1
```

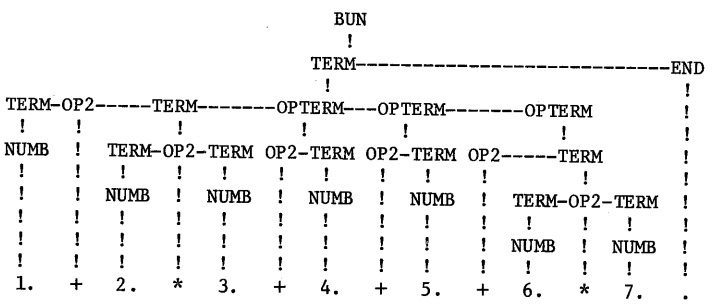


最終構文解析木

```
Kotae wa -4. desu.

Eval Sem-PART Time=18.ms  G.C.Time=0.ms
$ 1+2*3+4+5+6*7. ↵

Parsing Time=499.ms  G.C.Time=126.ms
PAGE1
```



不定数回繰り返し指定が活かされている。

```
Kotae wa 58. desu.

Eval Sem-Part Time=13.ms  G.C.Time=0.ms
$ Y :=MOD(X,13)+ABS(X-130). ↵

Parsing Time=1400.ms  G.C.Time=269.ms
PAGE1
                                    BUN
                                     !
 VAR-ASGN---------------------------TERM-----------------------------END
  !   !                               !                                 !
  !   !              TERM-------------OP2-----------TERM                !
  !   !                !              !              !                 !
  !   !   FUNC2-LK-TERM-COMMA-TERM-RK  !  FUNC1-LK-----TERM------RK     !
  !   !     !    !   !    !    !   !   !    !   !        !       !     !
  !   !     !    !  VAR   !   NUMB !   !    !   !  TERM-OP2-TERM !     !
  !   !     !    !   !    !    !   !   !    !   !    !   !   !   !     !
  !   !     !    !   !    !    !   !   !    !   !  VAR   !  NUMB !     !
  !   !     !    !   !    !    !   !   !    !   !   !    !   !   !     !
  Y   :=   MOD   (   X    ,   13.  )   +   ABS  (   X    -  130. )     .

Y ni 39. wo Dainyuu shimashita.

Eval Sem-Part Time=35.ms  G.C.Time=0.ms
```

4進木や6進木の活用により、構文解析木の構造が、分かり易いものになっている。

4.2 日本語

処理結果の例だけを示す。この例題では、活用形の判定にmessage中の情報を利用している。

```
? (HELLO)
BUN wo IRE te KUDASAI.
$ TAIHENMIMINOOOKINAZOUHA SORAWOTONDA. ↵

Parsing Time=1211.ms  G.C.Time=306.ms
PAGE1
                          BUN
                           !
                       SENTENCE-----------------------END
                           !                          !
                          VP                          !
                           !                          !
  ADV--------------ADV----------------ADV-----VERB     !
   !                !                  !       !      !
   !             NP----------JOSI   NP--JOSI VHT--AE   !
   !              !            !     !    !    !    !   !
   !       ADJ----ADJ---NOUN   !    NOUN  !    !    !   !
   !        !      !     !     !     !    !    !    !   !
   !     NOUN-NO   VP    !     !     !    !    !    !   !
   !      !   !    !     !     !     !    !    !    !   !
   !      !   !   VERB   !     !     !    !    !    !   !
   !      !   !    !     !     !     !    !    !    !   !
  TAIHEN MIMI NO OOKINA ZOU   HA   SORA  WO  TOND A    .
PAGE1
                          BUN
                           !
                       SENTENCE-----------------------END
                           !                          !
                          VP                          !
                           !                          !
                 ADV-----------------ADV-----VERB     !
                  !                  !       !      !
                NP--------------JOSI   NP--JOSI VHT--AE   !
                 !              !     !    !    !    !   !
           ADJ----------NOUN    !    NOUN  !    !    !   !
            !           !       !     !    !    !    !   !
```

文法にあいまいさがあったため同時に2つの最終構文解析木が出来あがった。

動詞の前の副詞句に対して不定数回繰り返し指定が活かされている。

次頁につづく。

```
          !              !    !    !    !    !    !    !
  ADV---NOUN-NO--VERB    !    !    !    !    !    !    !
   !     !   !    !      !    !    !    !    !    !    !
 TAIHEN MIMI NO OOKINA ZOU   HA   SORA  WO  TOND A    .

Eval Sem-Part Time=3.ms  G.C.Time=0.ms
$ ZOUGA TOBEBA MIMIMO TOBU. ↵

Parsing Time=268.ms  G.C.Time=0.ms
PAGE1
                    BUN
                     !
                SENTENCE----------------END
                     !                   !
          VP-------------SENTENCE         !
           !                  !           !
    ADV------VERB            VP          !
     !         !              !           !
   NP--JOSI V5H-V5T     ADV-----VERB      !
   !    !    !   !       !         !       !
  NOUN  !    !   !    NP--JOSI V5H-V5T    !
   !    !    !   !    !    !    !    !     !
   !    !    !   !  NOUN   !    !   !     !
   !    !    !   !    !    !    !   !     !
  ZOU   GA  TOB EBA MIMI   MO  TOB  U     .

Eval Sem-Part Time=2.ms  G.C.Time=0.ms
```

5 まとめ

拡張LINGOLにおいて文法規則が高々2進木までであったものをn進木に拡張し、また不定数回繰り返し機能を導入したことによって、更に柔軟な文法規則が可能となった。その結果、より少ない文法カテゴリーと文法規則を用いて、理解し易い文法大系を容易に作成、実行できるようになり、これを、簡単な数式と日本語の処理をもって確任した。しかも、その処理速度においても、従来のシステム（拡張LINGOL）と同等な性能を維持することができた。

なお本システムは、拡張LINGOLと同じ電総研LISP（LISP2.0）[4]上で作成した。作成に当り御協力頂いた淵一博氏をはじめとする電総研推論機構研究室の方々に深く感謝致します。

参考文献

1) Pratt V. R. : A Linguistic Oriented Programming Language, IJCAI3, 1973, 372-381.
2) Pratt V. R. : LINGOL - A Progress Report, IJCAI4, 1975, 422-428.
3) 電総研・推論機構研究室 ：拡張 LINGOL - 自然言語処理のためのプログラミングシステム，1978.
4) 電総研: LISP User's Manual, EPICS-5-ON-4, 1978.
5) 田中穂積：計算機による自然言語の意味処理に関する研究，電総研研究報告第797号，1979.