TPL(Talkable Programming Language) 文法の整備と構文解析について

福田, 晃
九州大学大学院総合理工学研究科情報システム学専攻

徳永, 浩二
沖電気株式会社 | 九州大学大学院総合理工学研究科情報システム学専攻

駒宮, 安男
九州大学大学院総合理工学研究科情報システム学専攻

https://doi.org/10.15017/17659

出版情報：九州大学大学院総合理工学報告. 8 (1), pp.71-78, 1986-07-25. 九州大学大学院総合理工学研究科
バージョン：
権利関係：

TPL (Talkable Programming Language) 文法の整備と構文解析について

福田　晃**・徳永浩二*・駒宮安男***

(昭和61年3月31日　受理)

Grammatical Improvement and Syntactic Analysis of TPL (Talkable Programming Language)

Akira FUKUDA, Hiroshi TOKUNAGA and Yasuo KOMAMIYA

TPL (Talkable Programming Language), which is proposed by Prof. Komamiya, one of the authors, is aimed to be a universal common language. And that is assumed to be used only in scientific and technological fields. This paper describes the improvement of TPL grammar and the syntactic analysis of TPL. A keynote of syntactic analysis is to make a word order flexible.

1. まえがき

TPL (Talkable Programming Language の略称) が著者の一人駒宮により提案され[1)2)]，昭和56年度から当情報組織研究室に於て，諸言語の調査，単語造語法，文法論などの研究が進められてきた．

本論文の目的は，これまでの研究の成果も考慮に入れ TPL 文法を整備し，さらに，TPL 文の構文解析を通じて，TPL 構文規則の柔軟性を確認し，かつ，現時点での TPL の問題点を探ることにある．

TPL は，科学技術上の分野に限って，自然言語的な働きと，プログラム言語的な働きの両面性を持つことを目的としている．前者としては，国際会議等での討論用言語，および科学技術上の論文の記述が，後者としては，データベースとのインターフェース，あるいは，人工知能における知識ベース構築用言語などが，代表的なものとして考えられる．また，前者からは国際共通語としての役割が強調され，後者からは述語論理との対応が要求されよう．

本論文での TPL の文法を TTG-Ⅲ (Tentative TPL Grammar Ⅲの略) と名付ける．ここでの文法とは，単語の構造，品詞，構文規則までを含む．

TPL の文法の骨格は，古瀬の TTG[3)]，竹内の TTG-Ⅱ[4)] で決定している．TTG は認知心理学の観点から，TTG-Ⅱは計算機処理の観点から，それぞれ構築されている．それに対し，TTG-Ⅲでは，TTG，TTG-Ⅱを基本にしながらも設計の観点を，TPL を使う人間の立場に移した．そして，TTG-Ⅱで提案された拡張構文規則に特に注目し，それをさらに柔軟にして，喋りやすさと表現力を高めることに努めた．以下に TPL の設計方針をまとめて記す．

i) 多くの国々の人にとって発音し易いものでなければならない．
ii) 一字一音，一音一字とする．
iii) 文字，単語などは英語を基本として構成する．
iv) 多義語は絶対に作らない．
v) 語順に関する規則がかなり柔軟であること．
vi) 述語論理を基盤とすること．
vii) 造語の規則を決めておく必要がある．
viii) 不特定話者，連続音声認識が容易であることを目標とする．

2. 品詞について

ここでは，TTG-Ⅱと比べて，新たに決定した事項や変更した事項について述べる．

2.1. 限定詞[5)]

限定詞は，名詞の前に置いてその名詞が示す集合の要素を限定するときに用いられる．英語において，「すべての～」を表す every や，「ある～」を表す a が

*情報システム学専攻修士課程（現在　沖電気㈱）
**情報システム学専攻
***元情報システム学専攻

典型的な限定詞といえる．TPL の限定詞を **Table 1** に示す．

自然言語では，限定詞を含んだ文が数通りの解釈を持つということが生じる．例えば，

「すべての学生は，ある本を読む．」

という文は，述語論理に変換すると全称記号と存在記号の順序の違いにより以下の２通りが考えられる．

i) $\exists y\ \forall x\ \{\text{Book}(y) \cdot (\text{Student}(x) \rightarrow \text{Read}(x, y))\}$

ii) $\forall x\ \exists y\ \{\text{Student}(x) \rightarrow (\text{Book}(y) \cdot \text{Read}(x, y))\}$

これらの意味の違いを明確に表すために TPL では全称記号を “remo” と “ziruto” の二種類用意する．即ち，全称記号が後についた文では $\forall x$ を “remo x”，全称記号が前についた文では，$\forall x$ を “ziruto x” として区別する．こうすることによって，語順を制限せずに二つの異なった論理式を曖昧なく表現できる．上記 i)，ii) の述語論理式は TPL では以下のように表現される．

i) TPL: Remo Rarunogeru ridoke do riko Bukosa.

ii) TPL: Ziruto Rarunogeru ridoke do riko Bukosa.

また全称記号の意味が曖昧でない場合は “remo” を用いる．

(例) 述語論理：$\forall x \forall y\ \{\text{Student}(x) \rightarrow (\text{Book}(y) \rightarrow \text{Read}(x, y))\}$

TPL: Remo Rarunogeru ridoke do remo Bukosa.

限定詞を高々２個含む文は，会話でもよく使用されリアルタイムでその意味の違いを理解できると思われるが，限定詞が３個以上になるとリアルタイムで厳密な意味を使い分けることは不可能に近い．よって限定詞を３個以上含む場合は述語論理式を先に想定し，それから TPL に訳す．この場合，特別に語順に制約を設けることで，それらの意味の違いを厳密に区別する．語順は述語論理式において限定詞が現れる順とする．

Table 1　Determiner.

TPL	Meaning	Quantifier	Remarks
remo	all, every	$\forall$	$\exists y \forall x, \forall x \forall y$
ziruto	any(in Eng.)	$\forall$	$\forall x \exists y$
riko	a (in Eng.)	$\exists$	
niremo	niro＋remo	$\sim\forall$	
niziruto	niro＋ziruto	$\sim\forall$	
niriko	niro＋riko	$\sim\exists$	

(例)「全ての先生が全ての学生にある問題を教える．」上記文に対応する述語論理式として次のものが考えられる．

$\forall x \forall z \exists y\ [\text{Teacher}(x) \rightarrow \{\text{Student}(z) \rightarrow (\text{Problem}(y) \cdot \text{Teach}(x, y, z))\}]$

$\Leftrightarrow \forall x\ [\text{Teacher}(x) \rightarrow \forall z \exists y\ \{\text{Student}(z) \rightarrow (\text{Problem}(y) \cdot \text{Teach}(x, y, z))\}]$

述語論理式をみると，限定詞の順番は $\forall x$, $\forall z$, $\exists y$ となっている．故に，TPL 文は $\forall x$ にあたる “Remo Titogera” が最初に，$\forall z$ にあたる “to remo Rarunogeru” が次に，$\exists y$ にあたる “do riko Puraburemosa” が最後にという語順になる．従って，TPL 文は以下のようになる．

TPL: Remo Titogera titoke to remo Rarunogeru do riko Puraburemosa.

次に否定記号を含む述語論理式の TPL 表現について述べる．述語論理式で述語の直前に否定記号が付いている場合には，TPL 文においてその述語に対応する動詞の直前に，否定詞 “niro” を付けて表現する．また，述語論理式で限定記号の直前に否定記号が付いていれば，TPL 文においてその限定記号に対応する限定詞を否定の限定詞に換える．以上のことを次の例で説明する．

(例)　「すべての学生は勉強しない」　…①

「勉強する学生は存在しない」　…②

上記①②は同じ意味を表し，対応する述語論理式は以下となる．

述語論理式：$\forall x\ (\text{Student}(x) \rightarrow \sim\text{Study}(x))$ …①

$\Leftrightarrow \sim\exists x\ (\text{Student}(x) \cdot \text{Study}(x))$ …②

まず，上記の述語論理式①において，否定記号～を取り除いた述語論理式

$\forall x\ (\text{Student}(x) \rightarrow \text{Study}(x))$　…①′

に対応する TPL 文は，

TPL: Remo Rarunogeru sutadoke.　…①′

である．更に，述語論理式①は述語 Study (x) の直前に否定記号が付いているので，述語論理式①に対応する TPL 文は述語 Study (x) に対応する動詞 sutadoke の直前に否定詞 “niro” を付ければよい．従って，TPL 文は以下となる．

TPL: Remo Rarunogeru niro sutadoke.　…①

一方，上述の述語論理式②において，否定記号～を取り除いた述語論理式

$$\exists x\,(\mathrm{Student}\,(x)\cdot \mathrm{Study}\,(x)) \qquad \cdots ②'$$

に対応する TPL 文は

TPL: Riko Rarunogeru sutadoke.　　…②′

である．更に，述語論理式②は限定詞 ∃x の直前に否定記号が付いているので，述語論理式②に対応する TPL 文は，限定記号 ∃x に対応する限定詞 "riko" を否定の限定詞 "niriko" (**Table 1** 参照) に換えればよい．従って，TPL 文は以下となる．

TPL: Niriko Rarunogeru sutadoke.　　…②

2.2. 動　　詞

TTG-Ⅱ[4] では，動詞は以下に示す状態動詞と状態変化動詞の二種類に大別されていた．

- 状態動詞：継続している状態を記述する動詞であり TPL では品詞語尾 "oka" を用いる．
- 状態変化動詞：状態動詞で示される状態間の遷移を記述する動詞であり，TPL では，品詞語尾 "oke" を用いる．

TTG-Ⅲでは，継続している状態や進行している状態を記述する場合には，状態動詞 (品詞語尾 "oka") を用いて，瞬間的な動きの場合も時間的に長さがある動きの場合にも，動作自体を言及したい場合には，動作動詞 (品詞語尾 "oke") を用いる[6]．

- 状態動詞：継続している状態を記述する動詞であり TPL では品詞語尾 "oka" を用いる．
  (例　nafoka：知っている)
- 動作動詞：「時間的に長さのある動き」または「ある状態から別の機能への移行，あるいは変化」を記述する動詞であり，TPL では品詞語尾 "oke" を用いる．
  (例　nafoke：知る)

2.3. 前置詞の形容詞的用法

TTG-Ⅱでは，前置詞の形容詞的用法は採用せず，関係詞に変換して表現していた．しかし，この表現法に制限すると，使用する者は非常に不自由に感じる．そこで TTG-Ⅲでは，前置詞の形容詞的用法を採用する．英語の場合，前置詞を形容詞的に使うときは **Fig. 1** の形をしている．TTG-Ⅲでも，**Fig. 1** と同じ形を取り，この部分に関しては，語順を一意にする．また，TPL では，副詞的用法と形容詞的用法の前置詞を共通には使えない．かといって，別々に単語を用意したのでは，前置詞の数が増大し，覚えやすさ，文字列資源の両面で都合が悪い．そこで，TTG-Ⅲでは，接頭辞を使って用法の違いを区別する．

前置詞を形容詞的に使うときは，副詞的用法の前置詞の前に "ken" をつける．ただし，b, p, m の前では "kem" とする．

注) 格前置詞には，形容詞的用法は許さない．

(例) nebabo「～の上に (接して)」
→ kennebabo「～の上の」
mo「～の中に (静止)」→ kemmo「～の中の」
temo「～の中へ (動作)」
→ kentemo「～の中への」

3. TPL の構文解析

3.1. 解析の概要

入力から出力までの解析の概要を **Fig. 2** に示す．ここでの入力とは，音声認識部での認識結果，すなわち，子音グループ[7][8] にまで認識された子音表記 C と母音表記 V の連続 "C V C V ･･･" である．出力は，終端記号である TPL 単語と非終端記号である文法カテゴリーを木構造にしたものである．なお，解析のアルゴリズムは，一部文献 9) で紹介されている質問応答システムを参考にした．

次節以降で，**Fig. 2** の各処理について述べていく．

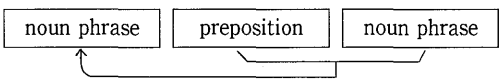

Fig. 1　Adjectival use of preposition.

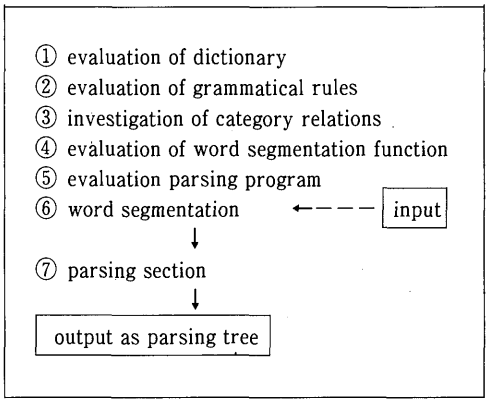

Fig. 2　Syntactic analysis of TPL.

3.2. 単語のセグメンテーション

入力文字列から単語間のセグメンテーションができるためには，TPL 単語の語幹が検出できればよい．そこで，セグメンテーションは2段階に分けて行う．

(1) 第一段階 (仮セグメンテーション)

TPL では，語尾の先頭に "o" が置かれ，語幹には "o" が含まれない．第一段階では入力文字列を "o" で分割し，分割された文字列を仮語幹とする．

(2) 第二段階 (主セグメンテーション)

第一段階で分割された各々の仮語幹を以下の方法により辞書で調べる．

i) 仮語幹が辞書内容により特殊語幹 (語尾として "o" だけを伴う語幹)[7][8] であることが判明すれば，その仮語幹は TPL における語幹であることが判る．

ii) 仮語幹が特殊語幹でなければ，"o" と次の単語の先頭2文字を結合したものを語尾として調べ，品詞を決定する．

iii) 他品詞に派生した単語の処理を行う．

iv) 仮語幹が辞書にないときは，接頭辞の可能性を調べる．

この段階までで単語も接頭辞も切り出せない場合は，入力の誤りと見なされる．

(単語分割の例)

"**MABOMASOPABOXAZO**"

→ ((N MAGO) (V NAFOKA) (P_{DO} DO) (N RAZO))

[但し，N：名詞，V：動詞，P_{DO}：対格前置詞]

3.3. 接頭辞の認識方法

接頭辞の種類を **Table 2**[10][11] に示す．接頭辞の認識方法を以下に示す．

接頭辞の切り出しは単語のセグメンテーションの処理内で行う．単語のセグメンテーションの第2段階で仮語幹が辞書内に存在しない場合に，以下のような接頭辞切り出しの処理を行う．

(接頭辞切り出しの例)

仮語幹 "**PEM・PEM**" が語幹辞書内に無かったとする．["**M**・" は TPL では撥音を表す．]

- まず，仮語幹を "**PE**", "**M・PEM**" に分割する．
- 次に，"**PE**" を接頭辞辞書で調べるが，無いので，仮語幹の分割を "**PEM**・", "**PEM**" とする．
- "**PEM**・" は接頭辞辞書にあるので接頭辞とする．残りの文字列 "**PEM**" は他の仮語幹と同様に処理する．

Table 2　Prefix.

TPL	Meaning
bante	Antonym of an adjective or an adverb
pere	Passive of a verb
bera	Inceptive aspect
zare	Perfective aspect
kubara	Same degree
mera	Comparative degree
masuta	Superative degree
ken(kem)	Adjectival use of a preposition

"**PEM**・" は形容詞的用法の前置詞を表す接頭辞 "KEN" の子音グループ表記であり，"**PEM**" は「～の中へ」という意味の前置詞 "TEMO" の語幹 "TEM" の子音グループ表記である．結局，仮語幹 "PEM・PEM" は次のように分割される．

(・・・, "**PEM・PEM**", ・・・)

→ (・・・, (PF KEN), (P_{ADV} TEMO), ・・・)

[但し，PF：接頭辞，P_{ADV}：副詞的用法前置詞]

3.4. 辞書の構造

辞書の構造が，単語とその意味を併記した構造を持つと，辞書引の際，辞書を端から順に逐一検索しなければならず時間がかかってしまう．

TPL の語幹は，全て子音で始まり，子音グループは **B** (有声破裂音)，**P** (無声破裂音)，**S** (無声摩擦音)，**Z** (有声摩擦音)，**M** (鼻音)，**X** (その他) である．そこで，TPL では辞書の構造を木構造にし，その最も高いレベルの構造を次のようにする．

辞書：(($\underline{\textbf{B} \cdots}_a$) (**P** ・・・) (**M** ・・・) (**S** ・・・) (**X** ・・・) (**Z** ・・・))

また，TPL は CVCV・・・という CV 音節の連続であるので，上で表した構造の1つ下のレベルの構造は，すべて母音で始まる．語幹には "o" を含まないので，結局，A, I, U, E の4つで始まる (例外として **M** の次は "・" を含むため5つとなる)．従って，上記下線部 a は次の構造を持つ．

部分辞書 (下線部 a)：

(**B** ($\underline{\text{A} \cdots}_b$) (I ・・・) (U ・・・) (E ・・・))

以下，その内部も再帰的に以上と同様の構造を持つ．

部分辞書 (下線部 b)：

(A (**B** …) (**P** …) (**M** …) (**S** …) (**X** …) (**Z** …))

実際の TPL の辞書は **Fig. 3** のように構成される．**Fig. 3** で「*」は辞書を構成する木構造の根を表す．また，NIL は空記号で，これは辞書の構造を再帰的にするために挿入される．辞書の構造をこのようにすると，検索回数は極端に少なくなる．

例として，"**BABEB**" なる語幹を辞書引する場合を考える．

- まず，辞書のいちばん上のレベルで，先頭が "**B**" のものを探す．この検索は，子音グループの個数が6個しかないので最悪の場合でも6回で終わる．
- 次に，部分辞書 (上で見つかった先頭が "**B**" の構造) の中で，先頭が "A" のものを探す．この検索は，母音の個数が4個しかないので最悪の場合でも4回で終わる．
- 以下，残りの "**B**", "E", "**B**" についても同様に検索すると最も多い場合でも6＋4＋6＋4＋6＝26回で "**BABEB**" の辞書項目に辿り着く．

一方，語幹ごとに登録する方法の検索回数は，最悪の場合は登録語数に等しい (800 語ならば 800 回)．

3.5. 構文規則の登録

構文規則の登録について例を示しながら述べる．

(例) 次の3つの構文規則を登録する場合を考える．

① NP：→N

② NP：→ZARATO S ZERATO

③ NP：→DET NP

[NP：名詞句　N：名詞　S：文　DET：限定詞
ZARATO～ZERATO：名詞節
また "：→" は書換規則の矢印を表す．]

- まず，各カテゴリーに『down』『up』『anc』という3つの属性を設定する (初期値は NIL)．各属性の定義は次の通りである．

『down』：自分を左辺に持つ構文規則群
『up』　：自分を右辺の先頭に持つ構文規則群
『anc』　：自分が生長して到達する可能性のあるカテゴリー群 (自分自身も含む)

＜『anc』属性の登録＞

構文規則に現れる全てのカテゴリーに対して，到達する可能性のあるカテゴリーを全て調べ (『up』属性を利用する)，その『anc』属性に登録する．

①～③の構文規則について上の処理を行うと，各カテゴリーの属性値は **Fig. 4** のようになる．

3.6. 構文解析部

ここでは，例として次の入力文が解析される様子を紹介する．但し，構文規則は文献 12) を参照されたい．

入力文："**MABOPASUPOXIBOPA**
BOBUPOSAPEMMOBESUPOSA："

入力文は，まず次のように単語単位に分割される．

単語群：((N MAGO) (TNS PASUTO) (V RIDOKA)
(P_{DO} DO) (N BUKOSA)
(P_{ADJ} KEMMO) (N DESUKOSA))
[但し，TNS：時制詞]

この TPL 文は英語と日本語でそれぞれ次のことを表現したものである．

英　語：I read the book in the desk.

日本語：私は机の中の本を読んだ．

さて，構文解析部の入力としては，上の単語群を用いるわけだが，厳密に言えば，各単語の品詞に基づいて解析木を作る．品詞だけを抜き出すと以下のようになり，ここから話を進めていく．

品詞群：(N TNS V P_{DO} N P_{ADJ} N)

```
(* NIL   (B <meaning of B>
        [A NIL (B <meaning of B A B> [A…] …)
                    :
                (Z <meaning of B A Z> [A…] …)
          :
        [E …]          )
        (P <meaning of  P> …)
        (M <meaning of  M> …)
        (S <meaning of  S> …)
        (X <meaning of  X> …)
        (Z <meaning of  Z> …) )
```

Fig. 3　Structure of dictionary.

NP	down (①②③) up (NIL) anc (NIL)	N	down (NIL) up (①) anc (NP)
ZARATO	down (NIL) up (②) anc (NP)	DET	down (NIL) up (③) anc (NP)

Fig. 4　Entry of rules.

TPL では，文の最も上層のレベルでは語順が自由である．このレベルに含まれるカテゴリーを次のようにリスト (SLIST と名付ける) にする．

SLIST：(NP VP PP_{DO} PP_{TO} PP_{ADV} ADV)

①まず，品詞群の先頭 N が辿り着く可能性のあるカテゴリーを SLIST から探し，それを目標 (ゴールと呼ぶ) として部分木を生長させる．いま，N の anc 属性に NP が含まれているので，N が NP に辿り着けることが判る．そこで，NP を目標として，N と NP を結ぶことを目指す (**Fig. 5- (a)** 参照)．

②次に，N の 1 つ上のカテゴリー (すなわち親) を決める．これは，N の up 属性を参照する．N の up 属性は，

N. up: [NP: →N, NP: →ADJ NP, …]

である．先頭の規則から調べる．各規則の左辺のカテゴリーに注目し，もしそれが目標カテゴリー (今は NP) に到達する可能性があれば，それを親とする (この例の場合は，先頭の規則が適用でき，NP を N の親とする)．(**Fig. 5- (b)** 参照)

③更に，N と並列な関係にあるカテゴリー (すなわち兄弟) を探す．上の②で適用した「NP: →N」を再び調べるわけだが，N の右にはカテゴリーはなく，N は単独で NP に生長することが判る (**Fig. 5- (b)** 参照)．

④ここで部分木の最上層のカテゴリーが NP になり目標と一致するが，結合する前に部分木の横の関係を調べる．N の次の品詞 (例では TNS) が，NP: →NP ? という規則の ? の部分に到達できるかどうかを調べる．到達の可能性がある場合は，? の部分を新しい目標として処理を進める．到達の可能性がない場合は，目標が達成されたことになり，次の処理に移る (**Fig. 5- (c)** 参照)．

⑤最初の部分木が完成したので，次の目標を決める．次の品詞 TNS について，再び②からの処理を行うと **Fig. 6** のようになる．

⑥次の目標は，品詞 P_{DO} から決める．以降の処理を **Fig. 7** に示す．

⑦結局，NP, VP, PP_{DO} の 3 つの大きな部分木が作られたわけだが，ここで，これらの部分木が，文のトップレベルの要素としての条件を満たしているかどうかを判定する処理について述べておく．つまり，

S: → {(NP) VP (ADV)* (PP_{do}) (PP_{to}) (PP_c) (PP_{adv})* }

[但し，{ } 内の語順は任意，( ) は省略可
( )*は任意個可]

の規則との照合である．これは，SLIST を調べることにより行う．大きな部分木が完成するたびに，その先頭のカテゴリーが PP_{ADV} と ADV 以外のものならば SLIST から取り除く．従って，上の例の場合は，SLIST は次のように遷移する．

SLIST: (NP VP PP_{do} PP_{to} PP_c PP_{adv} ADV)
→ (VP PP_{do} PP_{to} PP_c PP_{adv} ADV)
→ (PP_{do} PP_{to} PP_c PP_{adv} ADV)
→ (PP_{to} PP_c PP_{adv} ADV)

このようにすると，文のトップレベルに NP, VP, PP_{do}, PP_{to}, PP_c は1回しか出現できず，PP_{adv}, ADV は任意個出現できることになり上の構文規則に適合する．また，最終的な SLIST 内に VP が残っている場合には，入力文中に動詞句が存在しなかったわけだから，その入力は文としては不適格とみなす．上の例の場合は文として適格であり，最終的に

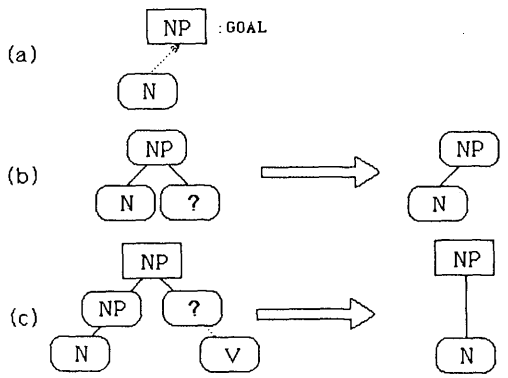

Fig. 5　NP-parse tree.

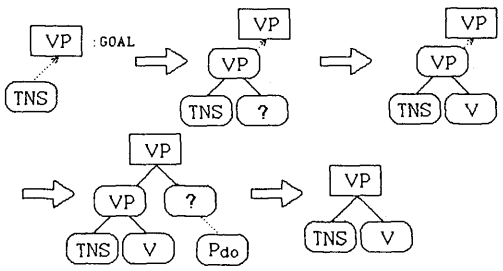

Fig. 6　VP-parse tree.

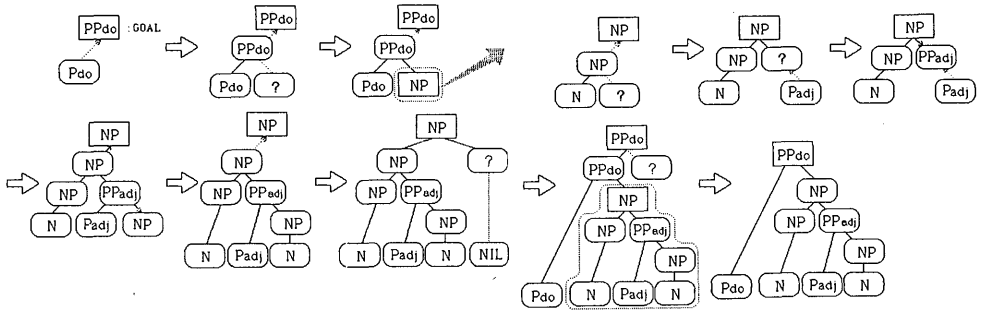

Fig. 7　PP_{do}-parse tree.

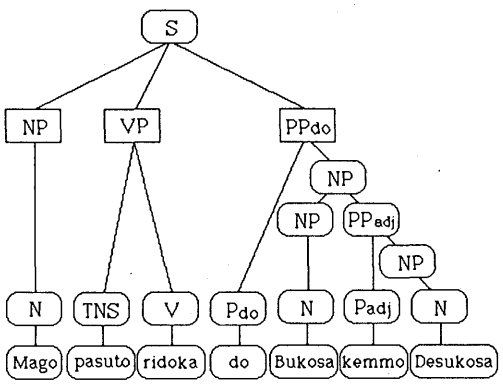

Fig. 8　S-parse tree.

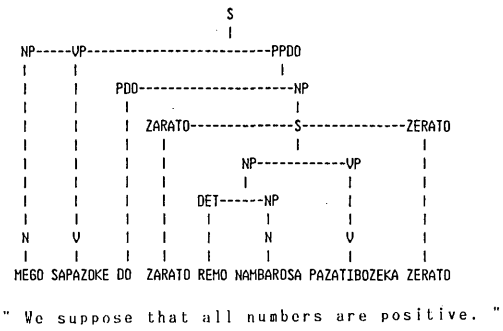

Fig. 9　Example of parsing tree.

Fig. 8 のような解析木が作られる．

4. 実 験 結 果

前節までのアルゴリズムを基に，TPL の構文解析プログラムを作成した．そのプログラムに実際に入力文を与えて解析を行った結果を **Fig. 9** に示す．

この実験で使用した文法には，疑問文·命令文に関連する文法は含まれておらず，ごく簡単な規則がうまく作用するかどうかを調べるにとどまった．また，文は単文のみを対象とした．だが，構文規則は非常に柔軟であり規則の追加を行うだけで拡張可能になると思われる．

5. む　す　び

TPL の構文解析を中心に述べた．簡単な文だけを対象とした実験であったが，語順自由の規則が実現できた．以下に TPL の現時点での問題点をまとめて示す．

- 固有名詞の取扱い
  固有名詞は，語尾をつけたり，母音 "o" を除いたりという TPL の方法に合わせるわけにはいかない．
- 専門用語の取扱い
  単語の構造を操作する点では問題ない．ただし，科学技術上の様々な分野で，それぞれ異なった意味で使われている専門用語の語義をどれか1つに特定することは難しい．

謝　　辞

本論文の作成にあたり，多大な御協力を頂いた本学情報組織研の修論生，卒論生諸氏に感謝の意を表します．

文　　献

1) 駒宮安男："「しゃべれるプログラミング言語」の提案", 信学誌, Vol. 62, No. 12 (1979).
2) 駒宮安男："「しゃべれるプログラミング言語」の提案 (その2)", 信学誌, Vol. 66, No. 6 (1983).
3) 古瀬蔵："「TPL の文法に関する認知科学の観点からの考察", 九州大学総合理工学研究科昭和 58 年度修士論文 (1984).
4) 竹内克志："TPL の文法の設計と図形問題証明システムの試作", 九州大学総合理工学研究科昭和 59 年度修士論文 (1985).
5) 前川浩司："TPL 文法の拡張と整備Ⅰ", 九州大学情報工学科昭和 60 年度卒業論文 (1986).
6) 中谷英一："TPL 文法の拡張と整備Ⅱ", 九州大学情報工学科昭和 60 年度卒業論文 (1986).
7) 福田, 小川, 徳永, 駒宮："TPL (Talkable Programming Language) (Ⅰ)", 情報処理学会, 自然言語処理研究会, 46−7 (1984).
8) 福田, 小川, 徳永, 駒宮："TPL (Talkable Programming Language) の概要と単語について", 九州大学総合理工学研究科報告, Vol. 7, No. 1 (1985).
9) 田中穂積他："LISP で学ぶ認知心理学3", 東大出版会 (1983).
10) 竹内, 近藤, 古瀬, 福田, 駒宮："TPL (Talkable Programming Language) (Ⅱ)", 情報処理学会, 自然言語処理研究会, 46−8 (1984).
11) 福田, 竹内, 駒宮："TPL (Talkable Programming Language) の文法について", 九州大学総合理工学研究科報告, Vol. 7, No. 1 (1985).
12) 徳永浩二："Talkable Programming Language の文法と構文解析に関する研究", 九州大学総合理工学研究科昭和 60 年度修士論文 (1986).