人工知能学会誌 Vol. 3 No. 1 (Jan. 1988) (別刷)

論 文

Technical Paper

Thai Syntax Analysis Based on GPSG

Vises Vorasucha* Hozumi Tanaka*

* Dept. of Computer Science, Faculty of Engineering, Tokyo Institute of Technology, Tokyo 152 Japan.

1987年4月21日 受理

Keywords: GPSG, ID and LP rule, DCG, BUP, Syntax Analysis.

Summary

In the process of doing syntax analysis of the Thai language, we found that GPSG can treat Thai syntactic rules in a very simple and compact way using Immediate Dominance (ID) and Linear Precedence (LP) rule. We built a simple GPSG translator which translates rules written in ID and LP rule format into DCG format. We call the translator 'Translator for ID and LP rule', or simply TIL. Besides the general translating function that is defined in GPSG, TIL provides several special formats which enable users to control the translation at the syntactic rule level. We also show that by using this translator we can reduce the rules to be handled by half.

1. Introduction

Computational natural language analysis has become increasingly important of late. Although languages like English, Japanese and a few others have been successfully researched, the Thai language remains untouched. Foreseeing a bigger role for natural language analysis in the future, we started this research.

We tried to do syntax analysis of Thai, and found that GPSG [1]-[3] can treat Thai syntactic rules in a very simple and compact way by using ID and LP rules, which are basic concepts of GPSG. To further the research, we built a simple GPSG translator which translates rules written in ID and LP rule format into DCG [8] format. We call this translator 'Translator for ID and LP rules', or simply TIL. Besides the general translating function defined in GPSG, TIL provides several special formats which enable users to control the translation at the syntactic rule level.

We write Thai syntactic rules in ID and LP rule format then use TIL to translate them into DCG rules which are finally translated into BUP clauses by BUP translator [4][5]. This is to prevent the left-recursive problems.

2. Thai Language and Thai Grammar

The following are some characteristics of the Thai language.

(i) No inflection at all. Instead of inflection, adjectives and adverbs are used to show tense and number.

(ii) Classifiers, that is the word signifying the unit of quantity of countable things, are used as in Japanese and Chinese. The classifiers are always put after the figures that show quantity.

(iii) Adverbs and adjectives are usually put after the words they modify.

(iv) Like in English, in Thai, the subject and the object are normally decided by the position in which they appear. A typical Thai sentence will look like pattern (2.1).

(2.1) S→subj(adj)verb(obj(adj))(adv).

Thai sentences are similar to English sentences in the sense that they begin with a subject, which is followed by a verb and an object. The differences occur with adjectives. In English, adjectives are placed before words they modify, but

in Thai, as we have mentioned above, they are placed after them.

According to Thai linguists [8][7][11], complete affirmative single sentences without an embedded sentence have only 12 patterns as shown in (2.2).

(2.2) s1→n.
s1→vp.
s1→n n.
s1→vp subj.
s1→subj vp.
s1→vp objd.
s1→vp objd obji.
s1→subj vp objd.
s1→objd subj vp.
s1→obji subj vp objd.
s1→subj vp objd obji.
s1→objd subj vp obji.

Now, let us consider the last three patterns. All of them have the same nonterminal symbols, but the only difference is that they are at different positions, i. e. they have different ordering. That is, the subject, object and verb in Thai sentences are, to a degree, movable in the sentences. In this aspect, Thai is similar to Japanese. Moreover, if we consider some of the nonterminal symbols to be optional, we can see that most of the rules in (2.2) are similar. Therefore, these rules can easily be classified into the two characteristics of immediate dominance and linear precedence. This hypothesis suggests that Thai syntactic rules may be expressed more compactly in ID and LP rule format than in normal format of context free phrase structure grammar.

The above does not only happen at sentence level, but it also occurs at other phrase levels in Thai. (2.3) is an example of Thai noun phrase rules, which shows that all of the rules produce less than four kinds of non-terminal symbols.

(2.3) np→mn.
np→mn num.
np→mn sbv.
np→mn df.
np→mn sbv num.
np→mn num sbv.
np→mn df sbv.
np→mn df num.
np→mn num df.
np→mn sbv num df.
np→mn num sbv df.
np→mn sbv df num.

Especially for prepositional phrases, they can occur at the beginning or end of a sentence without changing the meaning of the sentence at all. (2.4) shows the rule for this kind of sentence.

(2.4) s→pp s1.
s→s1 pp.

If there are more than one prepositional phrase, the number of possible combinations will become enormous. For example, (2.5) shows the case of a sentence which has three prepositional phrases.

(2.5) s→pp pt ps s1.
s→pp pt s1 ps.
s→pp ps s1 pt.
s→pt ps s1 pp.
⋮ ⋮
⋮ ⋮
s→ps s1 pp pt.
s→pt s1 pp ps.
s→pp s1 pt ps.
s→s1 pp pt ps.

We can easily represent these rules with only one ID rule without any LP rule, as shown in (2.6). Brackets mean that the non-terminal symbols enclosed are optional.

(2.6) s→s1, (ps), (pp), (pt).

If we use format of normal context free structure grammar, we will have to write about 50 rules for all of the rules that (2.6) represents.

3. Translator for ID and LP rules : TIL

To advance our research, we have built a simple GPSG translator, TIL, which translates rules written in ID and LP rule format into DCG format. And then use the translated DCG format rules to do syntax analysis.

3·1 Basic Mechanism of TIL

TIL is a translator written in Prolog. Basically, TIL will translate rules written in ID and

LP rule format into DCG rule format. ID rules and LP rules must be written in separate files in advance. Here, we are going to explain in more detail about how TIL translates those ID and LP rules into DCG rules (Fig. 1).

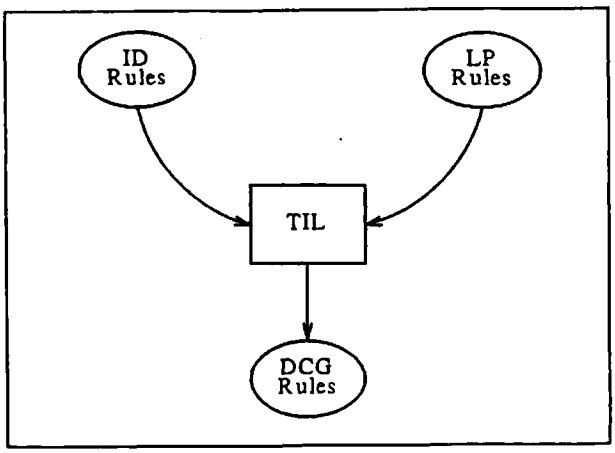

Fig. 1 Basic Mechanism of TIL.

〔1〕 *The format of ID and LP rules for TIL*

The normal format of ID rules for TIL is similar to the format provided in standard GPSG, except that there must be a period at the end of each rule. This is due to the restriction in Prolog. The format of ID rules for TIL will look like this.

(3.1) ID rule :

a→b, c, d.

b→a, c, d.

For LP rules, we also need a period at the end of each rule. Moreover, in standard GPSG, a symbol '<' is used to show the LP relation, but here, for TIL, we use symbol '<<' instead of '<'. This is because Prolog reserves the symbol '<' to be a less-than relation symbol. Then the format for LP rules will look like this.

(3.2) LP rule :

c << d.

〔2〕 *Representation of LP rules in the database*

When TIL starts the translation, TIL will read the LP rule file first. The information gained from each of the LP rules will be transformed into n_precd clauses.

(3.3) ID rules :

c << d.

(3.4) n_precd (d, c).

This is to be read as "d cannot precede c". The number of n_precd clauses will be equal to (n*(n-1))/2, where n is the number of non-terminal symbols in each of the LP rules.

〔3〕 *The Translation of TIL*

After all of the LP rules are read, TIL starts to look up the ID rule file. ID rules are not loaded into the database, but are dealt with in turn. When an ID rule is processed, a combination of non-terminal symbols of the body, the right hand side part of the mark '→', of that rule will be produced. The combination is checked to see whether it is against LP rules or not. If one of the n_precd clauses is successful, it means that the combination of the ID rule is against an LP rule, and cannot be used. Only those that do not match any of the n_precd clauses will be put to the output file. This process is repeated until the end of ID rule file.

The algorithm is roughly shown in Fig. 2.

For instance, TIL will translate the rule in

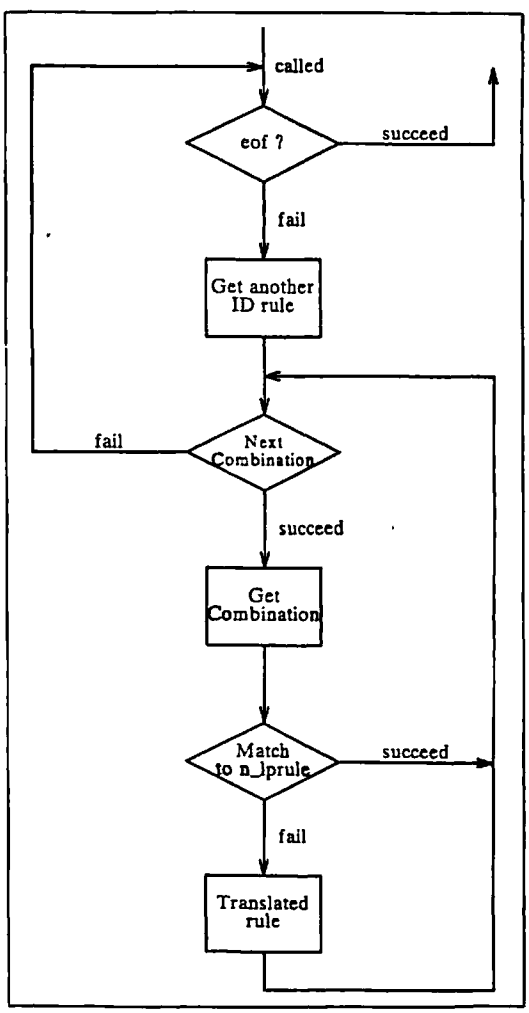

Fig. 2 The Translation of TIL.

(3.5) into (3.6).

(3.5) ID rule :

a→b, c, d.

b→a, c, d.

LP rule :

c << d.

(3.6) Translated Rule :

a→b, c, d.

a→c, b, d.

a→c, d, b.

b→a, c, d.

b→c, a, d.

b→c, d, a.

3·2 Translating Function of TIL

We have explained the basic mechanism of TIL and showed how TIL translates ID and LP rules into DCG rules. In this section, we are going to explain TIL's function.

[1] *Normal Translation*

Now, TIL does not support the metarule convention and feature instantiation principles, but can deal with a simple feature system. This is carried out by attaching one variable to each of the non-terminal symbols, and representing the feature system with list-structured data. Therefore, every non-terminal symbol must have one variable attached.

In fact, TIL will translate (3.5) into (3.7), instead of into (3.6).

(3.7) Translated Rule :

a(Ad)→b(Ad1), c(Ad2), d(Ad3).

a(Ad)→c(Ad1), b(Ad2), d(Ad3).

a(Ad)→c(Ad1), d(Ad2), b(Ad3).

b(Ad)→a(Ad1), c(Ad2), d(Ad3).

b(Ad)→c(Ad1), a(Ad2), d(Ad3).

b(Ad)→c(Ad1), d(Ad2), a(Ad3).

To make TIL more practical, we provide some more variants of formats besides format of the ID and LP rule that we have mentioned above. We will explain each of these formats in the sections following.

[2] *Parameter Defining Format*

Usually, information transmission from a non-terminal symbol to another non-terminal symbol is necessary. For instance, the number agreement of subject and verb phrase. This problem can be solved, if we can write into the syntactic rules something like (3.8).

(3.8) s→subj (NMB), vp (NMB).

TIL can manage this kind of rules. Users can define their own variables in any non-terminal symbols. TIL will not attach any variables to those non-terminal symbols, but keep the names of the variables in the non-terminal symbols as users have defined them. Moreover, instead of being a variable, for TIL, it may be a list-structured constant. Therefore, TIL will translate rules like (3.9) into (3.10).

(3.9) ID rule :

a→b, c (Adc), d (Adc).

b→a ([3]), c, d.

LP rule :

c << d.

(3.10) Translated Rule :

a(Ad)→b(Ad1), c(Adc), d(Adc).

a(Ad)→c(Adc), b(Ad2), d(Adc).

a(Ad)→c(Adc), d(Adc), b(Ad3).

b(Ad)→a([3]), c(Ad2), d(Ad3).

b(Ad)→c(Ad1), a([3]), d(Ad3).

b(Ad)→c(Ad1), d(Ad2), a([3]).

For LP rules, users can define a list-structured constant for any non-terminal symbols in LP rules. But this is not allowed for variables. LP rules will look like (3.11).

(3.11) LP rule :

subj<<vp([3])<<vp([4])<<obj.

[3] *Non-terminal Symbols Optional Format*

For ID rule, non-terminal symbols that are put between parentheses are taken to be optional by TIL. A rule in (3.12) is equivalent to two rules in (3.13) under TIL translation.

(3.12) ID rule :

a→b, (c, d).

(3.13) ID rule :

a→b, c, d.

a→b.

[4] *Augmentation Adding Format*

In each ID rule for TIL, we can add Prolog programs to the rule. This is done by enclosing the program with a left brace '{' and a right brace '}'. Now, Prolog program enclosed between

braces will be put to the end of the translated rule unconditionally. An example of this kind of rule is shown in (3.14) and (3.15).

(3.14) ID rule :
```
a→b (Adb), {Adb== [2]}, c.
LP rule :
b << c.
```
(3.15) Translated Rule :
```
a(Ad)→b(Adb), c(Ad2),
        {Adb== [2]}.
```

[5] *Direct Description Format*

The last format of TIL is provided for a user to write a syntactic rule directly in DCG format in the case that it cannot be written in ID and LP rule format.

In an ID rule file, clauses beginning with the mark '+' will be added to the translated rule file without being changed. (3.16) shows the format for this kind of rule.

(3.16) ID rule :
```
+bgst (Adb)→obji (Ado1),
            subj (Ads2),
            vp (Adv),
            objd (Ado4),
            {Adv== [4]}.
```

4. Thai Syntax Analysis System

In this chapter, we will explain in whole, our Thai syntax analysis system, and how we use TIL in doing syntax analysis. Again, we will show that Thai syntactic rules can be written more compactly in ID and LP rule format than in a normal grammar description language, such as DCG.

4·1 The Representation of Thai Syntactic Rules

Some parts of Thai syntactic rules have been provided by Thai Linguists already [6][7][11]. We rewrote those rules into ID and LP rule format, and added some new syntactic rules to those parts that were not sufficient enough for syntax analysis.

In this section, we will show some examples of ID and LP rules of Thai syntactic rules. All of our syntactic rules can be found in [13].

First, for noun phrases we use ID and LP rules shown in (4.1). Adm is for checking the agreement of the classifier's type and noun.

(4.1) ID rule :
```
np→mn (Adm), (sbv), (num
    (Adm)).
np→mn(Adm), df, (num(Adm)).
np→mn(Adm), sbv, (num
    (Adm)), df.
+np (Adnp)→mn (Admn1),
        df (Addf2),
        sbv (Adsbv3).
LP rule :
mn << sbv.
mn << num.
mn << df.
sbv << df.
```

These rules will be translated into rules in DCG format as shown in (4.2). Here, for simplicity, we have omitted some unnecessary variables.

(4.2) DCG rule :
```
np→mn, sbv.
np→mn, df.
np→mn, df, sbv.
np→mn (Adm), num (Adm).
np→mn (Adm), sbv, num(Adm).
np→mn (Adm), num(Adm), sbv.
np→mn (Adm), df, num (Adm).
np→mn (Adm), num (Adm), df.
np→mn (Adm), sbv, num(Adm),
    df.
np→mn (Adm), num(Adm), sbv,
    df.
np→mn (Adm), sbv, df, num
    (Adm),
```

We have rules for sentences that have a single-object transitive verb as shown in (4.3). The augmentation of this rule is to check whether the given verb is a single-object transitive verb or not.

(4.3) ID rule :
```
bgst→(subj), vp (Adv), objd,
                {Adv== [3]}.
+bgst (Adbgst) objd (Adobjd),
```

subj (Adsubj),
vp (Adv),
{Adv== [3]}.

LP rule :
subj << vp << objd.

(4.3) represents DCG rules shown in (4.4).

(4.4) bgst→subj, vp ([3]), objd.
bgst→vp ([3]), objd.
bgst→objd, subj, vp.

By treating Thai syntactic rules in this way, we can largely reduce the number of rules. Now, we use 35 ID rules and 13 LP rules to represent 113 DCG rules. .

4·2 The system as a whole

TIL translates rules in ID and LP format into DCG format, and we mentioned that we use those translated DCG format rules to do syntax analysis. But, in fact, after we have the translated rules, instead of using those rules directly, we again translate them into BUP clauses by using the BUP translator[4][5]. This is to avoid the complication of the left-recursive rule.

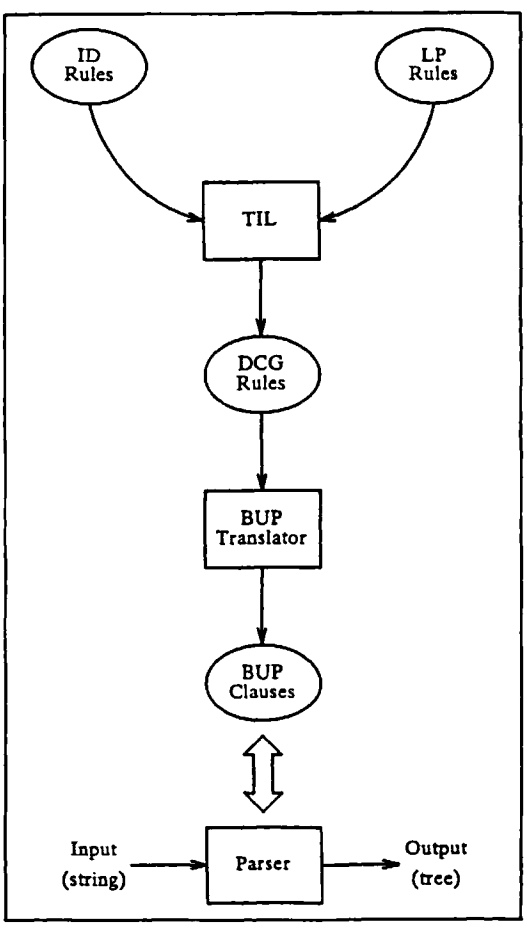

Fig. 3 The Outline of Thai Syntax Analysis System.

We do syntax analysis with BUP clauses. The input is a sentence of the Thai language, and the output is a tree-structured form of the components of the sentence. Therefore, the outline of our Thai syntax analysis system will look like Fig. 3.

5. Summaries and Topics

In the process of doing Thai syntax analysis, we found that, except for some differences like positions in which adjectives appear, Thai sentences are similar to English sentences in the sense that subject, verb and object have the same ordering. We also found that some components of Thai sentences can move around in the sentences to some extent without changing the meaning of the sentences. This characteristic is similar to Japanese.

We applied ID and LP rules, which are basic concepts of GPSG, to do Thai syntax analysis, and showed that Thai syntactic rules can be reduced by about half if we use ID and LP rule format instead of DCG format. Together with this, we built a basic GPSG translator : Translator for ID and LP rules (TIL). Needless to say, TIL can perform the normal translation of ID and LP rules. Moreover, TIL provides some special formats to give users a degree of control at the syntactic rule level.

In this chapter, we will assess our research and at the same time, touch on some topics related to our research.

5·1 Feature Instantiation Principles

We have not applied any feature instantiation principles to TIL yet. In fact, our syntactic categories have no internal structure at all. To define the structure, a through study of the Thai language to find out the kinds of information necessary for analysis, and to be included in the feature structure of the categories, is necessary. However, we have reserved one variable in each of the non-terminal symbols so that

feature conventions can be applied to TIL later.

To compensate for the lack of feature instantiation principles, we have utilized many kinds of formats to insure control in syntactic rules. We hope that users can use these kinds of format to perform the function of feature instantiation. This is largely because we have augmentation. By augmentation, a Prolog program is able to be accessed from syntactic rules, and the user can modify any controls with in the Prolog program. Until feature instantiation principles are applied to TIL, we hope that the user, in the meantime, will find the special formats of the present TIL helpful in the control aspects of syntactic rules. We will discuss more about the augmentation in the following section.

5・2 Metarule

In its current implementation, the Metarule convention has not been applied to TIL. The format for Metarule is not difficult, but there are some problems with the theory of Metarule. As mentioned by Pollard,

> First, ... as long as their application is suitably constrained, the grammars that they produce fall within context-free grammars. But if natural languages are not context-free, the advantage of metarules over transformations is no longer straightforwardly obvious. Second, work done in 1982 by Culy, and by Uzskoreit and Peters, has shown that a metagrammar using metarules with a single metavariable can generate infinite grammars for non-context-free languages... . And third, the devices which have been proposed so far for constraining the application of metarules have all been shown to have undesirable theoretical or empirical consequences [10].

Recently, there is a trend to do the analysis without Metarule or with some other devices instead of Metarule, like the Lexical Rule in Head-Driven Phrase Structure Grammar [9].

5・3 Augmentation

One of the TIL formats which is different from standard GPSG is an augmentation adding format. The augmentation adding format is an important tool to help us write syntactic rules more precisely and simply.

In this system, augmentations are replaced at the end of translated rules unconditionally. However, sometimes it is possible that augmentations are needed to be put not at the end of the rule, but between any specified syntactic phrases. We can do this by attaching to each augmentation in ID rules an identification number. Then, in LP rules, we can define the order of an augmentation as if it were a non-terminal symbol by using the identification number. The format will look like (5.1).

(5.1)
```
ID rule :
a→b(Ab),{1:Ab==[2],Ac== [3]},
   c (Ac), d.
LP rule :
b << {1:} << c.
```

Then, this will be translated into DCG format as shown in (5.2).

(5.2)
```
DCG rule :
a→b(Ab), {Ab== [2], Ac== [3]},
   c (Ac), d.
a→b(Ab), {Ab== [2], Ac== [3]},
   d, c (Ac).
a→b(Ab),d,{Ab==[2],Ac==[3]},
   c (Ac).
a→d,b(Ab),{Ab==[2],Ac==[3]},
   c (Ac).
```

If there is a need to put 'b', the augmentation and 'c' in a row without 'd' between them, this can be done by the simple technique of creating a new non-terminal symbol, as shown in (5.3).

(5.3)
```
ID rule :
a→new, d.
new→b (Ab), {1:Ab == [2],
      Ac == [3]}, c (Ac).
LP rule :
b << {1:} << c.
```

This produces rules shown in (5.4), which will perform what we want.

(5.4)
```
DCG rule :
a→ new, d.
a→ d, new.
```

new→b (Ab), {1:Ab == [2],
Ac == [3]}, c (Ac).

Using this kind of format, together with Direct Description Format, we would be able to provide more variations of control.

At present, the Thai syntactic rule set that we have is very small. They can process only simple sentences : sentences that have only one verb. One of our problems is that until now research on the Thai Language is very limited. To further the research of Thai syntax analysis, research on the Thai Language itself is indispensable.

Acknowledgement

We wish to thank Assistant Professor Noboru Akiyama in University of Electro-Communication ; Wilai Hongladaromp, a present staff member of Thai International Airways, for giving us a Thai character font program to support this research. And to Satoshi Konno, a present staff member of Toshiba Co. Ltd., for helping us to rewrite the program adapting it to the Sun 2 Workstation. Also many special thanks to students in Tanaka Laboratory for many helpful and valuable comments on the research and this paper, especially to Joseph C. Rouchar and Quek Chee Huei who spent their valuable time to correct the English in this paper.

◇ References ◇

(1) Gazdar, G. and Pullum, G. K. : *Generalized Phrase Structure Grammar : A Theoretical Synopsis*, Indiana University Linguistics Club, Indiana (1982).

(2) Gazdar, G., Klein, E., Pullum, G. K. and Sag, I. A. : "Coordinate Structure and Unbounded Dependencies," in *Developments in Generalized Phrase Structure Grammar : Standford Working Papers in Grammatical Theory*, ed. Ivan A. Sag, Vol. 2, pp. 38-68, Indiana University Linguistics Club, Indiana (1982).

(3) Gazdar, G., Klein, E., Pullum, G. K. and Sag, I. A. : *Generalized Phrased Structure Grammar*, Basil Blackwell, Oxford (1985).

(4) Matsumoto, Y., Tanaka, H., Hirakawa, H., Miyoshi, H. and Yasukawa, H., 1983 a : "BUP : A Bottom Parser Embedded in Prolog," *New Generation Computing*, Vol. 1, pp. 145-158 OHMSHA, LTD. and Springer-Verlag, Tokyo (1983).

(5) Matsumoto, Y., Masaki, K. and Tanaka, H., 1983 b : "BUP Translator-Automatic Generator of a Parser from a Grammer," *Bulletin of Electrotechnical Laboratory*, Vol. 47, No. 8, pp. 679-697, Agency of Industrial Science and Technology, Ministry of International Trade and Industry, Ibaraki, Japan (1983).

(6) Ngamsut,C.:ภาษาศาสตร์ภาษาไทยOdeon Store,Wang Burapha, Bangkok, in Thai (1981).

(7) Panupong, V. : *The Structure of Thai Grammatical System*, Ramkamhaeng University, Bangkok,in Thai (1984).

(8) Pereira, F. and Warren, D. : "Definite Clause Grammar for Language Analysis-A Survey of the Formalism and a Comparison with Augmented Transition Networks," *Artif. Intell.*, Vol. 13, pp. 231-278, North-Holland, Amsterdam (1980).

(9) Pollard,C. J. : "Generalized Phrase Structure Grammars, Head Grammar, and Natural Language," Ph. D. Dissertation, Standford University, Standford (1984).

(10) Pollard, C. J. : "Phrase Sturcture Grammar without Metarule," presented at the 1985 meeting of the West Coast Conference on Formal Linguistics, Los Angeles (1985).

(11) Thichinpong, P. : ลักษณะภาษาไทย Odeon Store,Wang Burapha, Bangkok, in Thai (1980).

(12) Vorasucha, V. and Tanaka, H. : "Thai Syntax Analysis Using GPSG," in *Proc. of The Second Conf. of Nihon Software Kagakkai : JSSST 1985*, pp. 45-48, Tokyo (1985).

(13) Vorasucha, V. : "Thai Syntax Analysis Based on GPSG," Master Dissertation, Tokyo Institute of Technology, Tokyo (1986).

〔担当編集委員：辻井　潤一〕

著者紹介

田中　穂積（正会員）

1964年東京工業大学理工学部制御工学科卒業. 1966年同大学院修士課程修了. 同年電気試験所（現，電子技術総合研究所）入所. 1983年東京工業大学工学部情報工学科助教授, 1984年同大学教授となり現在に至る. 工学博士. 人工知能，自然言語処理の研究に従事. 情報処理学会，電子情報通信学会，認知科学会，計量国語学会各会員.

Vises Vorasucha

1984年東京工業大学工学部電気電子工学科卒業. 1986年同大学院理工学研究科情報工学専攻修士課程修了. 現在，同大学院博士後期課程在学中. 自然言語処理，特に機械翻訳に興味を持つ. 情報処理学会，日本ソフトウェア科学会各会員.