Generation for Dialogue Translation Using Typed Feature Structure Unification

Yoshihiro UEDA
ATR Interpreting Telephony Research Labs.
Sampeidani, Inuidani, Seika-cho
Kyoto 619-02, Japan
ryu%atr-la.atr.co.jp@uunet.uu.net

Kiyoshi KOGURE
NTT Basic Research Laboratories
9-11, Midori-cho 3-chome, Musashino-shi
Tokyo 180, Japan
kogure%atom.ntt.jp@uunet.uu.net

Abstract

This article introduces a bidirectional grammar generation system called feature structure-directed generation, developed for a dialogue translation system. The system utilizes typed feature structures to control the top-down derivation in a declarative way. This generation system also uses disjunctive feature structures to reduce the number of copies of the derivation tree. The grammar for this generator is designed to properly generate the speaker's intention in a telephone dialogue.

1. Introduction

It is important for the generation part of the dialogue translation system to reflect the speaker's intention (illocutionary force). To properly translate the illocutionary forces, the Intention Translation Method has been developed at ATR (Kogure et al., 1989). This generator was developed as a part of the dialogue translation system.

Bidirectional grammar is helpful in maintaining the grammar/lexicon (Appelt, 1987). A feature structure representation has been adopted for analysis result and generation input because it can keep various information including illocutionary forces and pragmatics in a consistent way.

The infinite application of grammar rules is a common problem of the existing top-down unification-based generators (Shieber et al., 1989). The solution adopted here is to control the generation process by selecting appropriate rules to apply. Typed feature structures (Aït-Kaci, 1986) are utilized to describe the control of the generation process in a declarative way. This description can also be used to avoid the derivation of unnecessary trees and to increase the generation efficiency.

Another problem of the top-down generators is making multiple copies of the phrase structure when the generation process encounters multiple rule candidates. This problem can be treated by introducing disjunctive feature structures.

This article first describes the advantages of typed feature structures and disjunctive feature structures in sections 2 and 3. The grammar for the generation system and the generation results are shown in section 4. The current status of this project and future tasks are described in section 5.

2. Introducing Typed Feature Structures

2.1 Selecting Appropriate Rules

The basic mechanism of this generator is the top-down application of the grammar rules and construction of the feature structures of the daughter nodes.

It is important to avoid the derivation of unnecessary phrase structures by selecting appropriate rules to apply in order to increase the efficiency. Consider the following rules taken from D-PATR (Karttunen, 1986).

```
VP =HC*=> (VP XP)                                        (1)†
   (<!m sem cont> == <!head-dtr sem cont>)
   (<!head-dtr !subcat first> == <!comp-dtr-1>)
   (<!head-dtr !subcat rest> == <!m !subcat>))

VP =CH=> (VP PP)                                         (2)
   (<!m sem cont> == <!head-dtr sem cont>)
   (<!head-dtr !subcat first> == <!comp-dtr-1>)
   (<!head-dtr !subcat rest> == <!m !subcat>))
```

The construction of the semantic representations given to the mother nodes of these

† In this rule, =HC*=> link shows that the first element of the right hand symbols becomes the head daughter and the others the complement daughters. =CH=> link is also supplied for complement-head constructions. A symbol with an exclamation mark (!) indicates a predefined template. In this rule, !m stands for the mother, i.e., the left-hand VP.

two rules are the same (predicate-argument structure), as can be seen below:

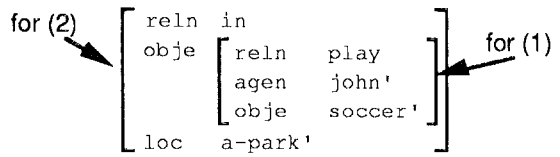

Fig.1 Sample feature structure

In generation, appropriate grammar rules must be selected using this representation. For this purpose, the difference between these feature structures must be found in the type of the key feature value's type. In this case, the reln (relation) feature plays the key role, and the value "play" must be of the verb type and "in" the prepositional type.

Typed feature structures formulated by Aït-Kaci (1986) are introduced to handle types in feature structures, because types cannot be handled by ordinary feature structure unification. Using typed feature structures, the following specifications can be attached to the former rules.

For (1): (<!m sem cont reln> == [*V*])
For (2): (<!m sem cont reln> == [*P*])

These specifications work as constraints to the rule application. The first specification shows that the <sem cont reln> feature of the node is unified to the type V (bracketed, shown in bold italics). If the unification succeeds, i.e., the <sem cont reln> feature is under V type in the type hierarchy, this grammar rule can be applied. The selection of appropriate grammar rules is thus accomplished in a declarative way.

2.2 Avoiding Termination Problem

There are various ways to utilize the type hierarchy. One example is subclassifying the categories.

One of the termination problems Shieber et al. (1989) pointed out is in the left-recursive rules. The rule (1) infinitely appends the subcat list to the daughter VP if the grammar is used for generation. This can be solved by restricting the permissible length of the subcat list††. The maximum length of the subcat list is 2, excluding the subject. This restriction can be represented as follows.

(:or (<!m !subcat> == [***list-end***]))
(<!m !subcat rest> == [***list-end***]))

However, this restriction forces the rule (2) to be applied twice to all verbs including intransitive verbs. Derivation of the phrase structures with incorrectly extended subcat lists will fail when the terminal is reached. This restriction can be solved more effectively using the type hierarchy. If verbs are classified into three subtypes (Monadic, Dyadic and Triadic) by the numbers of their arguments, the restriction in rule (1) can be written as follows.

(:or ((<!m sem cont reln> == [***dyadic***])
(<!m !subcat> == [***list-end***]))
((<!m sem cont reln> == [***triadic***])
(:or (<!m !subcat> == [***list-end***])
(<!m !subcat rest> == [***list-end***]))

2.3 Relating Types and Categories

Another function of the type hierarchy is using the types as a bridge between the semantics in the feature structure and the category in the CFG rules. Categories (nonterminal symbols) are also expressed by, and are closely related to, types. The following lexical entry definition shows that the complement of the verb is VP.

(deflex "must" dyadic
(<!subcat first> == [***VP***])
...)

If type VP is a subtype of XP, [XP] and [VP] are unified to bear [VP] when the lexical entry "must" is unified in rule (1). In D-PATR, such unspecified categories are treated by the system by introducing special symbols X, Y, etc. Typed feature structures serve as a sound foundation for this task.

3. Introducing Disjunctive Feature Structure Unification

Introduction of disjunctive feature structures solves the inefficiency caused by making copies of whole trees when a node can be applied to multiple candidates of rules.

For example, multiple copying is caused by the mutual restriction between the subject and the verb (subcategorizing by verb and subject-verb agreement). The verb cannot be determined until the subject is determined and the derivation tree must be copied for each verb candidate.

Instead of copying the derivation tree, the surface entries of a verb are packed into a

†† Though the restriction cannot be applied to languages like Dutch (Shieber et al., 1989), the limitation is irrelevant to our purpose (translation between Japanese and English).

disjunctive feature structure in a lexical entry as follows.

```
(DEFLEX-UNIT |be-Unit| DYADIC
  (:or  (!finite-form  !present-tense
        (:or    ((<word> == "am") !1sg-subj-agr)
                ((<word> == "are")
                 (:or ((!2sg-subj-agr) (!pl-subj-agr))))
                ((<word> == "is") !3sg-subj-agr)
        (!finite-form  !past-tense
        ......)
```

When the derivation proceeds and the subject is determined, one surface string is selected from these three candidates (see Fig. 2).

The unification of disjunctive feature structures is implemented according to Kasper's algorithm (Kasper, 1987).

4. Grammar and Examples

The grammar developed for this generation system is based on HPSG (Pollard and Sag, 1987) and its modification by Borsley (1987). Relating illocutionary forces to utterances is achieved in this grammar.

For example, consider the following feature structure including the REQUEST illocutionary force.

```
[CIRC[RELN [*REQUEST*]]
     [AGEN ?X03[IND-OBJ[LABEL  *SPEAKER*]]]
     [RECP ?X02[IND-OBJ[LABEL  *HEARER*]]]
     [OBJE [CIRC[RELN [*SEND-1*]]
                [AGEN ?X02]     ; *HEARER*
                [RECP ?X03]     ; *SPEAKER*
                [OBJE !a-reg-form']]]]
                      ; abbreviated here
```

From this feature structure, the following generation results can be obtained.

```
> (gen3 fs-1)
("would you send me a registration form"
"could you send me a registration form"
"send me a registration form")
;; "send NP to NP" form is suppressed here.
```

Specifying one of these results can be done by enriching the input feature structure.

5. Current Status and Further Tasks

This article described how the generation process is effectively controlled by typed feature structures and disjunctive feature structures.

The generation mechanism described here is implemented in Common Lisp on Symbolics Lisp Machines and Sun Workstations. A screen hardcopy of the environment is shown in Fig. 2.

The grammar for this generation system is now under enrichment. The relationships between surface utterances and intentions need to be further explored.

Kume et al. (1989) and Kogure et al. (1989) introduced illocutionary force type planning from deep illocutionary force type. Combining this method with the generator is the next task.

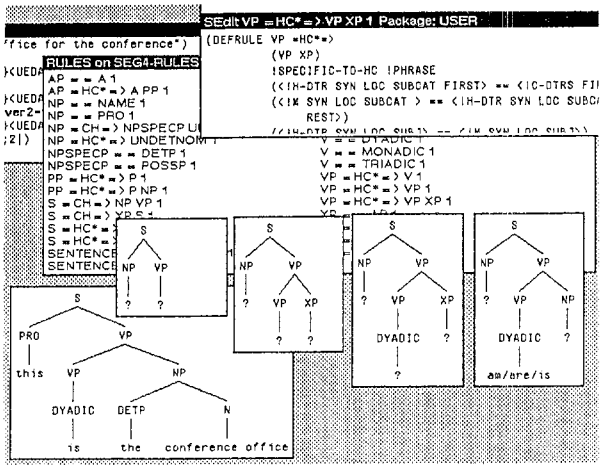

Fig.2 The Environment

Acknowledgement

The authors would like to express their appreciation to Mr. Hitoshi Iida and other researchers of ATR Natural Language Understanding Department for valuable suggestions and discussions.

References

Hassan Aït-Kaci (1986), An Algebraic Semantics Approach to the Effective Resolution of Type Equations, *Theoretical Computer Science 45*, pp. 293 - 351, North-Holland, Amsterdam.

Douglas E. Appelt (1987), Bidirectional Grammars and the Design of Natural Language Generation Systems, in *TINLAP-3*, pp. 185 - 191, Las Cruse.

Robert D. Borsley (1987), Subjects and Complements in HPSG, *Report No. CSLI-87-107*, CSLI, Stanford.

Lauri Karttunen (1986), D-PATR: A Development Environment for Unification-Based Grammars, Report No. CSLI-86-61, CSLI, Stanford.

Robert T. Kasper (1987), A Unification Method for Disjunctive Feature Descriptions, in *25th ACL*, pp. 235 - 242, Stanford.

Kiyoshi Kogure, Hitoshi Iida, Kei Yoshimoto and Teruaki Aizawa (1989), A New Paradigm of Dialogue Translation, in *International Symposium "Computer World '89"*, Osaka.

Masako Kume, Gayle K. Sato, and Kei Yoshimoto (1989), A Descriptive Framework for Translating Speaker's Meaning, in *4th ACL European Chapter*, pp. 264 - 271, Manchester.

Stuart M. Shieber, Gertjan van Noord, Robert C. Moore, and Fernando C. N. Pereira (1989), A Semantic-Head-Driven Generation Algorithm for Unification-Based Formalisms, in *27th ACL*, pp. 7 - 17, Vancouver.

Carl Pollard and Ivan A. Sag (1987), An Information-Based Syntax and Semantics, Volume 1, Fundamentals, *CSLI Lecture Notes Number 13*, CSLI, Stanford.