COLING 82, J. Horecký (ed.)
North-Holland Publishing Company
© Academia, 1982

ON A SEMANTIC MODEL FOR MULTI-LINGUAL PARAPHRASING

Kazunori Muraki

C & C Systems Res., Labs., Nippon Electric Co., Ltd.
1-1 Miyazaki, Yon-chome, Takatsu-ku
Kawasaki 213, JAPAN

The aim of the present paper is to formalize semantic-directed lexical selection by virtue of frame-based semantic inference capability built in the CFL representation language. The DG model of paraphrasing semantic descriptions can explicate logical process of knowledge-based sentence generation excluding any particular procedures for lexical selection or syntax structure generation. In addition this paper emphasises that this model is basically not dependent on target languages.

1. Introduction

This paper introduces a newly developed semantic-directed paraphrasing model, called DG (Declarative Generation) Model and also reports preliminary linguistic generation experiments done on system JASS (Japanese Synthesis System) developed under the DG framework.

While there already have been a few generation systems, such as BABEL(1)(N. M. Goldman), MUMBLE(2)(D. D. McDonald) etc. which intend to resolve lexical selection using special procedures or descrimination networks, the DG model can paraphrase semantic information written in the CFL frame language into target sentences using its built-in semantic inference capability. Conceptually, the model is divided into two logical phases, MU, which is the syntax generation phase, including lexical selection and syntax selection, and TLG, surface structure generation phase, including transformation and morphological generation. The model utilizes a semantic dictionary and a lexical dictionary both written in CFL for lexical selection, in which the functional requirements are limited to those semantic inference capabilities found in CFL. It is of great importance that these capabilities have already been used in the contextual understanding of languages.

The main DG model characteristics are as follow.
1) It presents a new way for semantic-directed lexical selection and syntax selection using frame inference capability. 2) Because of the modularity of all knowledge required for paraphrasing, this model greatly reduces knowledge base management costs. 3) It is generally independent of target languages, since the contents written in CFL and the built-in inference capability are thoroughly independent of languages. Accordingly, the DG logic functions are easily adaptable to the paraphrasing function for any natural language understanding systems or semantic-directed mechanical translation systems.

2. DG generation model basis

The main purpose behind proposing the DG paraphrasing model lies in formally bridging the qualitative gap between interlingua(4)(5)(6) and surface structures. The inputs are a sentence style indicator, a generation control, and a set of instance semantic depictions. In the first phase MU, input semantic depictions are transformed into a syntax structure utilizing a semantic dictionary, a lexical dictionary and syntax generation rules under the control of the sentence style indicator and the generation control. In the second phase TLG, the syntax structure is transformed into a surface string structure by a series of surface structure generation rules.

Contextually Factored Frame representation Language : CFL

CFL is a frame-based representation schema, with representation units called depictions correspond to both a dictionary entry and a semantic description. It has embedded semantic-directed inference capabilities, which function to transform input semantic descriptions to a structure with morphological and syntactic information.

Depictions describe sentential semantics based on Fillmore's case theory. Figure 1 shows the simplest examples of depictions. Here in the examples, depictions are classified into two categories; schema in Figure 1a and instance in Figure 1b. A schema is distinguished from an instance by the fact that all instance depictees (depiction names), except for distinct names, are postfixed by a distinct number. From a pragmatic viewpoint, schema compose a semantic dictionary in the long-term memory, while instances describe concrete events and descriptions in the short-term memory.

```
(*TRANS)                                 ((*TRANS 001)
 (DTYPE V CLASS)                          (DTYPE V IND)
 (INS D-(*TRANS 001)(*TRANS 002))         (INS          )
 (ako (a *LOCSTATECHANGE) with            (ako (a *TRANS) with
 (ACTOR     C (*PERSON))                  (ACTOR      V (*PETER  ))
 (TOPLACE   C (*PLACE))                   (INSTR      V (*CAR 001))
 (FROMPLACE C (*PLACE))                   (TOPLACE    C (*PLACE  ))
 (INSTR     C (*VEHICLE))))               (FROMPLACE C (*PLACE  ))))

                 a                                      b
```

Fig.1 Examples of semantic depictions

Fig.1a describes spacial-transportation (*TRANS) abstract where attribute descriptions are corresponding to case frame descriptions. Filling up ACTOR and INSTR with *PETER and *CAR .001, respectively, instantiates schema A and then produces the instance Fig.1b "Peter drives".

In CFL, two kinds of inference functions forms the basis for logico-semantic lexical selection.

Type 1. An implication test function acting on a combination of either an attribute value and an attribute condition, or one attribute condition and another attribute condition.
Type 2. An association test function between depictions.

Here, attribute condition is written by Boolean formula for semantic depictions. Type 2 function, which are realized by integrating Type 1 functions, can play a role in determining whether a semantic depiction is semantically identical to another or not. In natural language understanding, this facility is frequently used to determine referents.

Functional Description : FD

The FD schema, which is an n-ary tree structure, is used to describe syntax structures, syntax generation rules, and surface structure generation rules. Figure 7 shows a list-form representation of a FD structure, where the root node is DISCOURSE. The intermediate nodes in this framework are labeled with grammatical markers or case markers. The main merit of this kind of tree structure is that any value in leaf or substructure can be identified by its distinct path from the root node. Leaves are segregated into three kinds of values; string values, depictees and numerals.

The following two sections explicate the inherent mechanism along the line of the linguistic paraphrasing process.

3. Syntax Generation - its knowledge and processing

The MU mechanism is formalized by iterative invocation of two primitive operations; Match and Unify. The Match function adds morphological and local syntactic information to semantic depictions using a lexical dictionary and a semantic dictionary. After Matching, the Unify function is employed to modify and extend the given sentence style indicator (called intermediate syntax structure), by applying syntax generation rules to the structure obtained by Match. In the initial stage, Match is applied to the semantic depiction specified by a depictee in the FD sentence style indicator leaf (see Figure 4).

Match operation

Lexical depictions, the lexical dictionary entries themselves, feature a $-prefixed depictee, and play a primal role in the mapping of a semantic depiction into a morphological and syntactic structure. The lexical depiction format is basically equal to that of a semantic depiction, with some extensions. As shown in Figure 2, the attribute names in a lexical depiction have such forms as SUB (=ACTOR), TOLOC (=TOPLACE), VERB=LEX etc., which are divided into two categories: 1. X (=Y), 2. U=Z .

The following must necessarily hold for a successful Match of lexical and semantic depictions.

i) Y must exist as an attribute name in a semantic depiction.
ii) X=Y (transformed in Match), and U=Z must be partial paths in a final syntax structure.

Now, assume semantic depiction *A (for example *TRANS.001 in Fig.1) is given. The process first tries to find the lexical depiction $B (for example $DRIVE in Fig.2), one of whose ancestors has a depiction name (depictee) identical, except for their prefixes, to one of the ancestors for the given semantic depiction.

```
(($DRIVE)
 (INS )
 (ako (a $LOCSTATECHANGE) with
 (SUB     (=ACTOR     ) C (*PERSON))
 (TOLOC   (=TOPLACE   ) C (*PLACE))
 (FROMLOC(=FROMPLACE) C (*PLACE))
 (BY      (=INSTR     ) C (* CAR))
 (VERB=LEX    M Drive)
 (VERB=VOICE M Active)
 (VERB=REFL  M (drive drove driven))
     .
     .
     .                          ))
```

Fiq.2 A lexical depiction

```
(($DRIVE)
 (INS )
 (ako (a $LOCSTATECHANGE) with
 (SUB     (=ACTOR     ) ? (*PETER))
 (TOLOC   (=TOPLACE   ) C (*PLACE))
 (FROMLOC(=FROMPLACE) C (*PLACE))
 ( BY     (=INSTR     ) ? (*CAR 001))
 (VERB= LEX     M Drive  )
 (VERB= VOICE  M Active)
 (VERB= REFL   M (drive drove driven)
     .
     .
     .                          ))
```

Fig.3 A Match result

If no such lexical depictions are found, MU will terminate. If the depictions are found, the following steps will be taken.

First, for each X_i (=Y_i) attribute name, the attribute value or condition for attribute name Y_i in the semantic depiction is tested to determine whether it implies the attribute condition of X_i(=Y_i). If Y_i does not exist in the semantic depiction, or if the test fails, Match tries to find the next lexical depiction.

Each Y_i value or condition is set as the X_i (=Y_i) value or condition if and only if all X_i (=Y_i) attributes satisfy the above test, as well as the tests for all Y_i case markers in the semantic depiction are completed. Otherwise, Match continues to search for a suitable lexical depiction.

Second, for attribute Y_i in the semantic depiction, which is not tested by the above, Match adds newly MOD=Y_i attribute with the value or condition of Y_i .

Consequently, the result appears simultaneously into the selected lexical depiction. The lexical depiction $DRIVE in Fig.3 is an example of Match results, which comes from a semantic depiction *TRANS 001 in Fig.1b and a lexical depiction $ DRIVE in Fig.2. Thus, the Match result has morphological information about DRIVE and local information about the surface and semantic case structure induced by $ DRIVE.

Unify Operation

In general, a depictee under a path in a sentence style indicator or intermediate syntax structure, must be transformed to one comforming to legitimate syntax structures. A syntax structure generation rule determines such legitimate structures according to the condition along the path. Such permissable structures are plural, so Unify must select the one appropriate to the lexical depiction obtained by Match.

Figure 4 shows a simple sentence style indicator which specifies that an instance depictee *TRANS.001 must be transformed to the syntax structure appropriate to the path <DISCOURSE=SEG=EVENT> .

```
              ┌ CAT=D                   ┐
DISCOURSE =   │                         │
              └ SEG=EVENT?(* TRANS 001) ┘
```

Fig.4 A sentence style indicator

```
((EVENT=) ──> (((CAT=S) ((SUB=?)) (DOB=?)) ((IOB=?))
               ((LOC=?))  (TIMEP=?)) ((VERB=?))))
```

a

```
((!GR=!SR   (- !CASE) =)──>(((CAT=PP))
                             ((CASE=) (((CAT=POSP)) ((LEX=!CASE))))
                             ((POB=) (((CAT=NP)) ((MOD=?)) ((HEAD=?)))
                                     ((CAT=S) ((COMLEX=KOTO)) ((ACT=?)))))
```

b

Fig.5 Examples of syntax generation rules

The FD syntax generation rule is shown in Figure 5a. The rule specifies that an instance semantic depictee just below the partial path < EVENT > is able to have the syntax structure specified by the right hand side of the rule. Figure 5b is a slightly extended form although it has basically the same function as the former. This includes variables !GR, !SR and !CASE, each of which has a distinct domain. For example, !GR (Grammatical Roles) can bind an element of a set { SUB, DOB, IOB etc.} . !SR (Semantic Roles) can bind that of { ACTOR, OBJECT, INSTR etc. } . !CASE has a domain of Japanese postpositions { GA (surface CASE for SUB), WO (surface CASE for OBJECT etc. } .

The FD syntax generation rule means that an instance depictee specified by a partial path which is an instance of <!GR=!SR> can be transformed to the structure indicated by the right hand side of the rule, as long as the depictee is prefixed by (-!CASE). Such a variable !CASE, as on the right, is replaced by the value if the rule is successfully applied.

Now, assume a lexical depictee A obtained by Match under a path $< a_1 = a_2 = \ldots = a_n >$. Generally, generation rules $\{R_i\}$ exist with path specifications $\{ <aj= \ldots = a_n> \}$, $1 \leq j \leq n$. Unify fails if any R_i is not found. Here, each R_i candidate generation rule is to be unified with depiction A in turn, starting from the rule with the longest path specification until a sound generation rule is found. Successful Unify is defined as follows.

Let the attribute name set for depiction A be $B= \{ <b_1 = b_2 = \ldots = b_j> \}$, and the set for all partial paths in R_i rule be $C= \{ <c_1 = c_2 = \ldots = c_k> \}$.

i) For each $< b_1 = \ldots = bj >$, there exists a $< c_1 = c_2 \ldots = b_1 = \ldots = b_\ell > \in C$, $1 \leq \ell \leq j$. Each attribute value of $< b_1 = \ldots = bj >$ is set to the value of extended path $< c_1 = c_2 = \ldots = b_1 = \ldots = bj >$ (or equal to $< c_1 = c_2 = \ldots = c_k = b_{\ell+1} = \ldots = b_j >$).

ii) All attribute values in depiction A must be assigned to the appropriate paths.

If an R rule is verified unsatisfiable, a new one is tried. If no other candidate is found, the generation process fails and terminates.

Thus, given a sentence style indicator (or an intermediate syntax structure), Match is applied to a semantic depictee and Unify to the lexical depiction resulting from Match. After this one primitive cyle, a new intermediate syntax structure is produced, which has morphological and syntactic information in greater detail than the previous one. Application of these two primitive operations continues until instance semantic depictees disappear from intermediate syntax structure.

4. Surface Structure Generation - its knowledge and processing

The TLG model for surface structure generation[3] is defined on a set of pattern-directed production rules written in extended FD structures, each of which specifies a source structure on the left hand side and target structure on the right.

The transformations required for surface structure generation can be roughly classified into two sub-classes. One is for global transformations, such as voice-transformation, nominalization, adjectivation etc. Another is mainly for morphological generation concerning tense, inflexion, gender etc.. In general, these two sub-classes have inherent application ordering. This holds not only between the above two sub-classes, but also holds among the members of the former. To support flexible rule application controllability, the TLG model is defined on an adaptive production system, in which rules are categorized and rule application order is determined by tags of rules and categories.

Figure 6 exemplifies a voice-transformation rule. Any rule has a rule number, a matching pattern including variables, a Boolean formula, a pattern-program and a tag. Variables prefixed with $ or # in a FD matching pattern can bind a substructure or a path, respectively. The Boolean formula is the LISP S-expression with these variables in a matching pattern, with a value T signifying that the rule application conditions have been satisfied. A pattern program is basically an FD structure with embedded functions such as (FUNC ENT ($Z)). In this example, FUN indicates that ENT is one place function with $Z as an argument. The last part of any rule is a tag which is a pointer to subsequent target categories.

Additionally, there are also tags in each category. A rule tag will specify that the control jump to the category specified by the tag, if the rule is applied. Control goes to the next rule in the same category if a tag is nil or the rule fails. On the other hand, a category tag will specify that control jump to the category specified if none of rules in the category can be applied. If a control tag is nil and none of the rules in the category can be applied, control goes back to the caller.

```
       [CAT=D                             ]    [CAT=D                                  ]
       [SUB=$X                            ]    [SUB=$Y                                 ]
(No3) •[DOB=$Y                            ]·T ➧[DOB=$X                                 ]•((CAT NOi)
       [VERB=  [VOICEF=A-P TRANS]         ]    [VERB= [VOICE=PASSIVE                  ]]
       [       [VOICE =ACTIVE   ]         ]    [      [LEX= [PASS=be                 ]]]
       [       [LEX    =$Z      ]         ]    [      [     [PP=(FUNC ENT($Z))       ]]]
```

Fig.6 A voice-transformation rule

5. Sentence Style Selection

Sentence style selection is a most difficult problem in linguistic generation. In the DG model, input sentence style indicator, generation control and surface structure generation rules directly contribute to sentence style selection.

Sentence style indicator roughly guides the style into which input depictions are paraphrased by placing the instance semantic depictees in the FD structure values. Values other than instance semantic depictees also determine how these depictees are paraphrased by application of surface structure generation rules, because the values can influence invocation of these generation rules. Consequently, sentence style selection must be accomplished while satisfying the contextual requirement in paraphrasing.

Generation controls have several kinds of information, which in reality can be consulted by Boolean formulas, and embedded functions in surface structure generation rules. Accordingly, generation control, more precisely, surface structure generation rules, can play a great role in determining which sentence style will be selected. The Japanese Synthesis System selects sentence styles, for example, polite, rude, abbreviated style etc. Such indication is held in generation control, so rules appropriate to those destinations are easily selected in TLG phase.

6. Experiments with JASS (Japanese Synthesis System)

JASS has been developed in LISP, in which all dictionaries and rules are stored in a secondary storage. At present, knowledge which describes news regarding accident in newspapers is composed of about 160 semantic depictions and 130 lexical depictions. Rules are composed of 40 syntax generation rules and 50 surface structure generation rules. The latter are classified into 6 categories, which are GLOBal, CONJunctive, CORE, PHRASE, LOCAL and MORPHological. In addition, seven kinds of embedded function for surface structure generation rules are utilized, mainly for morphological generation.

In Figure 7, a portion of a simple example of a FD syntax structure obtained by the MU process of JASS generation system is given, and is transformed from the event wherein 34 SAI NO TAKUSHI UNTENSHU GA KUROI KURAUN WO NUSUMU is described, using an input description set. It means 34 year old taxi driver X steals a large black luxury car.

```
( (DISCOUSE =)
  ( (CAT =)
    ( (D)
      ( (SEG =)
        ( (EVENT =)
          ( ( (CAT =)
              ( (S)))
            ( (VERB =)
              ( ( (LEX M UBA))
                ( (INFLT M GODAN))
                ( (INFL M D-SHUSHI))))
            ( (SUB =)
              ( (ACTOR)
                ( ( (CAT =)
                    ( (PP)))
                  ( (CASE =)
                    ( ( (CAT =)
                        ( (POSP)))
                      ( (LEX =)
                        ( (GA)))))
                            .
                            .
                            .
                                      )
```

Fig.7 A portion of a syntax structure for an example sentence

7. Conclusion

DG verification has been successfully accomplished through experiments using JASS. The DG succeeded in constructing a knowledge-oriented as well as semantic-oriented paraphrasing model from semantic descriptions, free from syntactic and morphological information. A very important factor is that such functions as inference capability for CFL, as well as the adaptive production system are not thoroughly differentiated from the common functions in the AI field, but are extended slightly. These functions proposed in DG, are used in different forms, especially in the fields of natural language understanding research. In this sense, the DG paraphrasing method has a great effect on future semantic-directed paraphrasing systems and multi-lingual translation systems.

Acknowledgement

The author thanks research managers Dr. T. Mikami and Dr. K. Hakozaki for their support for this research, and research leader Mr. T. Makino for fruitful discussions.

References

1 Goldman, N. M., Sentence paraphrasing from a Conceptual base, CACM. 2 18 (1975) 96-166.

2 McDonald, D., Preliminary Report on a program for generating natural Language, IJCAI4 (1975) 401-405.

3 Nagao, M. et. al., On English generation for a Japanese-English translation system, Technical Report on Natural language processing of Information Processing of Japan 25 (1981).

4 Bobrow, D. G. et. al., An overview of KRL, a knowledge representation language, Studies in Cognitive Science 1 1 (1977) 3-46.

5 Schank, R. C. and Abelson, R., Scripts, Plans, Goals, and Understanding (Lawrene Erlbaum Associates, Hillsdale, New Jersey, 1977).

6 Ramelhart, D. E. and Norman, D. E., Active Semantic Networks as a Model of Human Memory, IJCAI3 (1973) 450-457.