A GRAMMAR USED FOR PARSING AND GENERATION

Jean-Marie LANCEL (*), François ROUSSELOT (**), Nathalie SIMONIN (*)

* CAP SOGETI INNOVATION, 129, rue de l'Université, 75007 - PARIS
** University of STRASBOURG II, 22, rue Descartes, 67084 - STRASBOURG

1. INTRODUCTION

This text presents the outline of a system using the same grammar for parsing and generating sentences in a given language. This system has been devised for a "multilingual document generation" project.

Martin KAY has shown that parsing and generation could be done using Functional Grammars. APPELT [APP85] and McKEOWN [McK82], among others, have used Functional Grammars for generation. Usually the grammar formalism is more suited to parsing than to generation. The way a grammar is processed for parsing is rather clear, while the generating process needs strong assumptions on the interpreter to be easily readable.

The Functional Grammar notation described here allows a full symmetry between parsing and generating. Such a grammar may be read easily from the point of view of the parsing and from the point of view of the generation. This allows to write only one grammar of a language, which minimizes the linguistic costs in a multilingual scheme.

Description of the Functional Grammar notation, in chapter 2, will thoroughly refer to Functional Descriptions and Functional Unification. For a detailed presentation, the reader may refer to [KAY79] [ROU84] [SIM85].

2. THE GRAMMAR FORMALISM

The formalism we have defined allows us to write a single grammar of a language which is used both for analysis and generation by means of two specialized interpreters.

Sentence analysis is viewed as the transition from a surface structure to a semantic representation which is a Functional Description. Sentence generation is the transformation of a semantic representation into a syntactic form. This symmetry between the two processes has to be clearly expressed if we want a clear notation, easy to read and to understand from the point of view of parsing and of generating.

A grammar rule is itself represented as a Functional Description. This FD has three main "identifiers" : PATTERN, MEANING and CONSTRAINTS.

Example of a simple grammar rule :

```
simple_gn =
 [ pattern = (det subst adj)
   meaning = [ obj = <subst meaning>
               definitude = <det type>
               qualif = <adj meaning>
               number = <subst number> ]
   constraints = ([equal = (<det gender>
                            <subst gender>
                            <adj gender>)
                   equal = (<det number>
                            <subst number>
                            <adj number>)])
   det     = [cat = det]
   subst   = [cat = subst]
   adj     = [cat = adj]                ]
```

The pattern part describes the syntactical structure. Each item of the list associated to pattern refers to a rule or to a terminal. In the above example the three terms refer to terminals. Omissions and repetitions are allowed.

The meaning part describes the semantic representation associated to the syntactical structure. Bracketed lists represent "paths" refering to Functional Descriptions inside the rule or in another rule. During parsing, these paths are used to build the semantic representation while in generation they are used for splitting a semantic structure into different sub-structures. The two processes, parsing and generation, are detailed in chapters 3 and 4.

The constraints part is a list of "set of constraints" expressed by Functional Descriptions. At least one "set of constraints" must be fulfilled. In the above example this allows us to express agreement rules used for both parsing and generating.

As in Martin Kay definitions a rule may have different derivations. These are represented by enclosed braces. Example :

```
simple_phrase = {
 pattern =      (gnl vtrans gn2)
 meaning =      [action =  <vtrans meaning>
                 subjsem = <gnl meaning>
                 objsem =  <gn2 meaning>]
 constraints = ([equal = (<gnl number>,
                          <vtrans number>])

 pattern =      (gnl vintrans)
 meaning =      [action =  <vintrans meaning>
                 subjsem = <gnl meaning>]
 constraints = ([equal = (<gnl number>,
                          <vintrans number>])
                }
```

3. THE PARSING PROCESS

3.1. Use of the grammar for parsing

In order to analyze a sentence, the words and compounds words are converted in Functional Descriptions, using a morphological analyzer and a dictionary. The result is a list of FD's which will be processed by the parser.

Example (semantic values are expressed here by English terms but they are usually expressed as FD) :

"les chaussures vertes" ("the green shoes")

Input list of parser is :

```
(
[cat = det      [cat = subst    [cat = adj
type =defined   gender = fem    gender = fem
number=plural   number=plural   number=plural
lex = le]       lex=chaussure   lex = vert
                meaning=shoe]   meaning=green]
)
```

This sentence matches with the rule simple_gn described in chapter 2, as the first FD of the list is functionnaly unifiable with [cat = det], the second FD with [cat = subst] and the third FD with [cat = adj].

The parsing process builds a structure which is a copy of the rule simple_gn and enlarges it with the actual word analyzed. The path descriptions are replaced by their actual values.

3.2. Structure built

```
simple_gn =
[ pattern =     (det subst adj)
  meaning =     [obj = shoe
                 definitude = defined
                 qualif = green
                 number = plural ]
  det       =   [cat = det
                 type = defined
                 number = plural
                 lex = le ]
  subst     =   [cat = subst
                 gender = fem
                 number = plural
                 lex = chaussure
                 meaning = shoe ]
  adj       =   [cat = adj
                 gender = fem
                 number = plural
                 lex = vert
                 meaning = green ]      ]
```

This structure is built if the constraints are met : for this rule it implies agreement of gender and number, which is the case for "les chaussures vertes".

4. THE GENERATING PROCESS

4.1. Use of the grammar for generation

The generation takes as input a semantic structure and produces a sentence.

As an example the rule simple_gn (cf chapter 2), is activated with the semantic structure

```
[ obj        = box
  definitude = undefined
  qualif     = white
  number     = plural ]
```

A copy of the rule is built. The paths in the Functional Description associated to the identifier "meaning" are used to convey the semantic information to the items referred to by the identifier "pattern" (These items are named "constituents").

In this example it gives :

identifier	path	pointed value
obj	<subst meaning>	box
definitude	<det type>	undefined
qualif	<adj meaning>	white
number	<subst number>	plural

The interpretation process of the grammar "builds" the path, which means that the needed identifiers are included in the copy of the rule.

```
FD for DET becomes :
     det  = [ cat = det
              type = undefined ]
where "type" has been added.
```

```
FD for SUBST becomes :
     subst = [ cat = subst
               meaning = box
               number  = plural ]
where "meaning" and "number" have been
added.
```

```
FD for ADJ becomes :
     adj = [ cat = adj
             meaning = white ]
where "meaning" has been added.
```

Then the constraints are applied. In the parsing process they are used to eliminate wrong constructions, while in the generating process they are used to transmit information.

Use of the constraints of the rule simple_gn

```
equal =
 (<det gender> <subst gender> <adj gender>)
```

At this step, this rule doesn't transmit any information because identifier "gender" is not present in at least one Functional Description

```
equal =
 (<det number> <subst number> <adjnumber>)
```

This rule transmits number of substantive (number = plural), in the two other Functional descriptions of the output list

After constraints are applied, the output list is :

```
  ([cat =     det
    type =    undefined
    number =  plural ]

   [cat = subst
    meaning = box
    number = plural ]

   [cat = adj
    meaning = white
    number = plural ])
```

The next step is word selection : for each terminal, the semantic structure associated with it is used to choose a lexical item. This is done by using Functional Unification. For each word or compound word selected, "constraints" are processed again, in order to transmit informations to Functional Descriptions of the list.

For a given structure there may be more than one adequate word. In that case the appropriate word is chosen by the user interactively.

The list of terminals is enlarged by the selected lexical items, as shown in the following example :

For the first item :

```
(
[cat=det        [cat = subst  [cat = adj
type=undefined  meaning=box    meaning=white
number=plural   number=plural] number=plural]
lex = un ]
)
```

For the second item :

```
(
[cat=det         [cat=subst    [cat=adj
 type=undefined   meaning=box   meaning=white
 number=plural    number=plural number=plural
 lex=un           lex=boite     gender=fem ]
 gender=fem ]     gender=fem ]
)
```

For the third item :

```
(
[cat=det         [cat=subst    [cat=adj
 type=undefined   meaning=box   meaning=white
 number=plural    number=plural number=plural
 lex=un           lex=boite     gender=fem
 gender=fem ]     gender=fem ]  lex=blanc ]
)
```

At this step each word of the output list is completely defined. The morphological generation processes each Functional Description using fields LEX, GENDER, NUMBER, MODE, TENSE and PERSON. The appropriate form of the lexical item is constructed using Functional Unification.

For this example the list constructed by the morphological generation is :

```
    ( "des",  "boites",  "blanches" )
```

which gives :

```
      "des boites blanches"
```

This example is a simple case where items of a "pattern" do not refer to other rules. Presence of a rule name in a pattern leads to activation of this rule with a subset of the initial meaning (transmitted by a path, as for a terminal).

4.2. Generation models

The generation of the sentence associated to a semantic structure may lead to various syntactical constructs. In order to reduce the number of constructs, and to allow control of text style, a specific feature has been introduced, named "generation model". A generation model associates a semantic pattern to a precise grammar rule.

Example :

Semantic structure associated to the advice "Do not expose to rain" in a user's manual :

```
[advice
 advice-type = directive
 advice-giver = constructor
 content      = [link = negation
                 argl = [action
                         action-type = expose
                         subjsem = user
                         objsem = machine
                         obj2 = rain ] ] ]
```

Among the "generation models" of the system, the following is Functionnaly Unifiable to the above structure :

```
[advice
 advice-type = directive
 gen-model = { [cat = prop-infinitive
                pattern = (gvinf *comp)
                meaning = <content> ]

               [cat = prop-must
                pattern = (gvdir *comp)
                meaning = <content> ]
             }                          ]
```

Remark : the symbol * means that the rule may be repeated.

This generation model is selected by a restricted version of Functional Unification : identifiers "advice" and "advice-type" must be present in the semantic structure.

In this example two grammar rules are candidate once the generation model is selected. A simple implementation is to choose a rule at random, another is to have an evaluation module which choose the most appropriate rule according to stylistic knowledge (technical style, telegraphic style, etc).

5. DEVELOPMENTS

Previous version of the multilingual generation system uses a grammar for parsing, and production rules for generation.

Present work is the adaptation of the parser to the new formalism, and the implementation of the generation interpreter. It includes the adaptation of the multilingual dictionary retrieval process.

6. REFERENCES

APPELT,D.E.
"Planning English Sentences." Cambridge University Press. 1985.

KAPLAN,R.M. and BRESNAN,J.
"Lexical-Functional Grammar : A Formal System for Grammatical Representation." In : Bresnan,J. (ed) The Mental Representation of Grammatical Relations. MIT Press. 1982.

KAY,M.
"Functional Grammar." Proceedings of Fifth Annual Meeting of the Berkeley Linguistics Society. 1979.

KAY,M.
"Unification Grammars." Xerox publication. 1981.

McKEOWN,K.
"Generating Natural Language Text in Response to Questions about Database Structure." Ph.D. dissertation. University of Pennsylvania. 1982.

RITCHIE,G.
"Simulating a Turing machine using functional unification grammar." ECAI 84. Pisa. 1984.

ROUSSELOT,F.
"Réalisation d'un programme comprenant des textes, en utilisant un formalisme unique pour représenter toutes les connaissances nécessaires." Thèse d'Etat. University of Paris VI. 1984.

ROUSSELOT,F. and GROSCOT,H.
"Un langage déclaratif uniforme et un analyseur syntaxico-sémantique." COGNITIVA 85. Paris. 1985.

SIMONIN,N.
"Utilisation d'une Expertise pour engendrer des textes structurés en français." Thèse. University of Paris VI. 1985.