Lexical-Functional Transfer: A Transfer Framework in a Machine Translation System Based on LFG

Ikuo KUDO
CSK Research Institute
3-22-17 Higashi-Ikebukuro, Toshima-ku, Tokyo, 170, Japan

Hirosato NOMURA
NTT Basic Research Laboratories
Musashino-shi, Tokyo, 180, Japan

Abstract

This paper presents a transfer framework called LFT (Lexical-functional Transfer) for a machine translation system based on LFG (Lexical-functional Grammar). The translation process consists of subprocesses of analysis, transfer and generation. We adopt the so called f-structures of LFG as the intermediate representations or interfaces between those subprocesses, thus the transfer process converts a source f-structure into a target f-structure. Since LFG is a grammatical framework for sentence structure analysis of one language, for the purpose, we propose a new framework for specifying transfer rules with LFG schemata, which incorporates corresponding lexical functions of two different languages into an equational representation. The transfer process, therefore, is to solve equations called target f-descriptions derived from the transfer rules applied to the source f-structure and then to produce a target f-structure.

1. Introduction

A grammatical theory called LFG (Lexical-functional Grammar)[1] is a framework for sentence structure analysis and has a simple framework for representing lexical and grammatical information. It analyzes a sentence in two steps, a phrase structure analysis and a functional structure analysis. The former is a syntactic analysis and produces constituent structures (c-structures). The latter consists of several procedures, attaching lexical functions to components in the c-structure, deriving functional equations called functional descriptions (f-descriptions) from them with preserving configurational relationships, and solving these equations to produce a functional structure (f-structure). Those lexical functions are represented by a representative framework called LFG schema.

We adopt such LFG schema to a representative framework for a dictionary and rules which define functional correspondences between components of two languages. With them the transfer process can be designed as a simple procedure such that its task is only to solve functional equations of the target language and then produce an f-structure of the target language. We propose such a framework called LFT (Lexical-functional Transfer). It consists of both a representative framework for a two-way dictionary and transfer rules and a processing mechanism of transferring an f-structure of source language into an f-structure of target language. The representative framework is declarative and then easy to manipulate. The procedure is a mathematical processing and thus enough simple and clear in its nature and executable easily.

2. Overall construction of translation system

Figure 1 shows the global construction of the translation system. The whole process is divided into three subprocesses, analysis, transfer and generation as usual translation systems. The analysis process is nothing but LFG analysis.

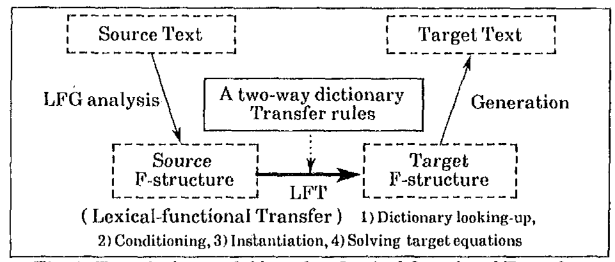

Fig. 1 Translation model based on Lexical-functional Transfer

The transfer process, LFT converts an f-structure of a source language into a corresponding f-structure of a target language. At first, a transfer dictionary is looked-up and transfer rules are selected. Next, the conditions in the rule are checked. If they are satisfied, the schemata of target language in the transfer rule are instantiated. And then the functional descriptions of target language are obtained. They are called the target functional descriptions (target f-descriptions). After setting up the target f-descriptions, the task of the transfer process is reduced to solve them and then produce an f-structure of the target language. The processes of instantiation and solving target f-descriptions are the same mechanism within LFG. Implementation and execution of these processes are very clear and thus there is no need for further explanation.

The generation process is tentatively defined as a linearization process of the structured relationships in the target f-structure and a insertion process of inflected words. However its explanation is beyond the scope of this paper.

3. LFT representative framework

3.1 Transfer rules

A transfer rule makes two schemata of two languages correspond each other and its general representative framework is as follows:

J[(LFG) schemata] < = = = > E[(LFG) schemata].

In the expression, to show what language the schemata belong to, a initial letter of each language is put in front of each square bracket. In this paper, Japanese is signified with 'J', English with 'E'. Examples of the transfer rules are as follows:

J [(↑ SUBJ)= ↓] < = = = > E [(↑ SUBJ) = ↓],

```
J[(↑PRED)='トム']<===>E[(↑PRED)='Tom'].
```

A metavariable ↑ in the right hand side must correspond to that in the left hand side, and also a metavariable ↓ in the right hand side must correspond to that in the left hand side. A symbol < = = = > designates that both sides are strictly corresponding. When a rule is referred in the transfer process, if it is, for example, transferring from Japanese into English, the side having 'J' plays like a condition part in a 'IF...THEN...' rule, and vice versa. Therefore the description of the transfer rules are bidirectional since both sides can be a condition part depending on the direction of transferring.

The number of schemata in both sides are not always equal and such an example appears in the rules 3 in the table 3. It can be divided into next three rules. The isolated type is used in a dictionary since it is compact.

```
E [(↑SUBJ)=↓                 ]<===>J [(↑SUBJ)=↓
                                      (↑SUBJ CASE-NAME)=SUBJ]
E [(↑OBJ)=↓                  ]<===>J [(↑OBJ)=↓
                                      (↑OBJ CASE-NAME)=OBJ  ]
E [(↑PRED)='PLAY             ]<===>J [(↑PRED)='suru
      <(↑SUBJ)(↑OBJ)>'                   <(↑SUBJ)(↑OBJ)>'
                                      (↑SUBJ CASE-MARKER)='ha'
                                      (↑OBJ CASE-MARKER)='wo']
```

In a f-structure, its structure is represented with hierarchy and function names. Even if the structures between two corresponding f-structures are different, a transfer process must prove well-formed syntactic relationships in the target f-structure. Even these relationships can be represented with the LFG schema. For example, the rule (2.c) makes different structures correspond; hierarchy and function names in the rule are different. English side is 'ACOMP SCOMP' but Japanese side is 'XCOMP'. Therefore LFT rule can make two different structures correspond.

Furthermore, there is often nothing corresponding between two languages. For example, a infinitive 'to' exists in English, but there is nothing in Japanese. Two schemata in the rule (2.b),

E[(↑ ACOMP SCOMP to) = +],

E[(↑ ACOMP SCOMP INF) = +],

represent infinitive 'to'. As another example, there is no gender in Japanese and English noun, but there are genders in French and German. But it is easy to treat the problem because you have only to add the gender's schema to the rule. For example, 'a book' in English corresponds to 'ein Buch' in German.

```
E [(↑PRED)='book'   ]<===>G [(↑PRED)='Buch'
   (↑NUM)=SG                 (↑NUM)=SG
   (↑SPEC)=A                 (↑SPEC)=EIN
                             (↑GENDER)=NEUTER]
```

3.2 Two-way dictionary

The LFT utilizes a two-way dictionary which has entries for both languages. Each entry consists of pairs of (1) a **designator** and (2) some **pointers**. The designator is a medium to instantiate the schemata in the condition side. The pointer refers a transfer rule. The rule is referred by both languages through each pointer.

A rule is registered to the 'value' entry of the head schemata, '(↑PRED)=value'. When a rule has many head schemata, it is assigned to all the 'value' entries redundantly. For example, the idiom 'be eager to' has two head schemata; (↑ PRED) = 'BE <...>', (↑ ACOMP PRED) = 'EAGER <...>' in the rule (2b). So it is assigned to the 'be' entry and the 'eager' entry in the table 1. But the designators are different. The 'be' designator is '↑ = ↓' and the 'eager' designator is '(↑ ACOMP)= ↓', as shown in table 1.

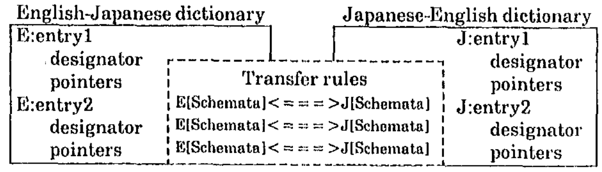

Fig.2 example of a two-way dictionary

Table 1 The English-Japanese dictionary (Ex. from sentence(1) to sentence (2))

English entry	desinator	pointers
be	↑ = ↓	rule2 a,b,c,d
eager	(↑ ACOMP)= ↓	rule2 a,b,c,d
Tom	↑ = ↓	rule1
play	↑ = ↓	rule3
baseball	↑ = ↓	rule4

Table 2 The Japanese-English dictionary (Ex. from sentence(2) to sentence (1))

Japanese entry	desinator	pointers
tagaru	↑ = ↓	rule2 a,b,c,d
tomu	↑ = ↓	rule1
suru	↑ = ↓	rule3
yakyuu	↑ = ↓	rule4

Table3 Transfer rules (Ex. between the sentence (1) and (2))

```
(Rule1)
E [(↑PRED)='Tom'             ]<===>J [(↑PRED)='Tomu'
   (↑NUM)=SG                          (↑NUM)=SG
   (↑PERSON)=3                        (↑PERSON)=3              ]
(Rule2.a)
E [(↑SUBJ)=↓                 ]<===>J [(↑SUBJ)=↓
                                      (↑SUBJ CASE-MARKER)='ha'
                                      (↑SUBJ CASE-NAME)=SUBJ   ]
(Rule2.b)
E [(↑ACOMP SUBJ)=(↑SUBJ)
   (↑ACOMP SCOMP SUBJ)
        =(↑ACOMP SUBJ)                [(↑XCOMP SUBJ)
   (↑ACOMP SCOMP to)=+        <===>J        =(↑SUBJ)
   (↑ACOMP SCOMP INF)=+                (↑PRED)='tagaru
   (↑ACOMP PRED)='EAGER                 <(↑SUBJ)(↑XCOMP)>'     ]
        <(↑SUBJ)(↑SCOMP)>'
   (↑PRED)='BE
        <(↑SUBJ)(↑ACOMP)>'   ]
(Rule2.c)
E [(↑ACOMP SCOMP)=↓          ]<===>J [(↑XCOMP)=↓               ]
(Rule2.d)
E [(↑TENSE)                  ]<===>J [(↑TENSE)=PRESENT
             =PRESENT                 (↑ASPECT)='iru'          ]
(Rule3)
E [(↑SUBJ)=↓                          [(↑SUBJ)=↓
                                       (↑SUBJ CASE-MARKER)='ha'
                                       (↑SUBJ CASE-NAME)=SUBJ
   (↑OBJ)=↓                   <===>J   (↑OBJ)=↓
                                       (↑OBJ CASE-MARKER)='wo'
   (↑PRED)='PLAY                       (↑OBJ CASE-NAME)=OBJ
        <(↑SUBJ)(↑OBJ)>'     ]         (↑PRED)='suru
                                          <(↑SUBJ)(↑OBJ)>'     ]
(Rule4)
E [(↑PRED)='baseball'        ]<===>J [(↑PRED)='yakyuu'
   (↑CAT)=SPORT                       (↑CAT)=SPORT             ]
```

4. LFT processing mechanism

LFT processing is divided into four phases as shown in Figure 3. Each phase is described briefly as follows:

(phase1) Looking-up the dictionary: Collect all the head f-descriptions whose type is '(f_n PRED)=value', from a source f-structure. Look-up 'value' in the dictionary one by one and go to the phase (2).

(phase2) Conditioning: Check whether the conditions in the rule are satisfied with the source f-structure. If so, go to the phase (3). If not, check the other rules. When a rule is applied (from English to Japanese), English side in the rule works the conditions , Japanese side works the result.

E [conditions] < = = = > J [results]

(phase3) Instantiation: Instantiate the schemata in the result side with **the table of correspondence**, which yields target f-descriptions. When actual variables (f_1, f_2, ..., etc.) are assigned to the metavariables ↑, ↓ in the results, the table is looked up. The table shows that actual

variables in the condition side correspond to that in the result side. For example, table 5 in the Figure 3.

(phase4) Solving target f-descriptions: After the phase (1), (2) and (3), collect all the target f-descriptions and solve them by the LFG algorithm, 'from f-descriptions to an f-structure'. So a target f-structure is obtained.

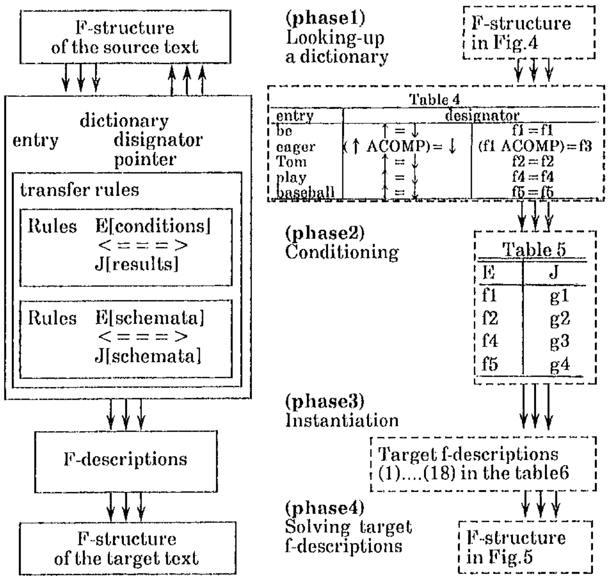

Fig.3 Mechanism of LFT (Lexical-functional Transfer)

During phases of 1, 2 and 3, metavariables are assigned to actual variables as follows:

(phase1) The metavariables $\uparrow$ or $\downarrow$ in the designator:
The '$\downarrow$-variable in the designator in the dictionary' is unified with the actual variable 'fn' in the schema '(fn PRED)=value' which is looked up. If designator is '$\uparrow = \downarrow$', assign the same variable 'fn' to '$\uparrow$-variable in the designator'. If not, assign the actual variable unified with the source f-structure. If it is not found, the conditions are not satisfied.

(phase2) The metavariables $\uparrow$ or $\downarrow$ in the condition side:
Assign 'actual variable which is assigned $\downarrow$-variable in the designator during (phase 1)' to '$\uparrow$-variable in the conditions'. Find the actual variables unified with the source f-structure. Assign it unified with the source f-structure to the $\downarrow$-variable. If it is not found, the conditions are not satisfied.

(phase3) The metavariables $\uparrow$ or $\downarrow$ in the result side:
Find the actual variables in the condition side by corresponding relations ($\uparrow$ to $\uparrow$, $\downarrow$ to $\downarrow$) which the rule define. Look up the variable in the table of correspondence. Assign the variable to the metavariable. If there is no variable, assign a new actual variable to the metavariable.

5. Example

An English example sentence and its Japanese equivalent sentence are as follows:

(1). *Tom is eager to play baseball.*
(2). トム は 野球 を し たがっている。
tomu ha yakyuu wo si(suru) tagatteiru(tagaru).

The f-structure of the English sentence is shown in Figure 4, and the f-structure of the Japanese sentence is shown in Figure 5.

(1) Collect all the f-descriptions '(f_n PRED)=value's from a source f-structure (Figure 4).

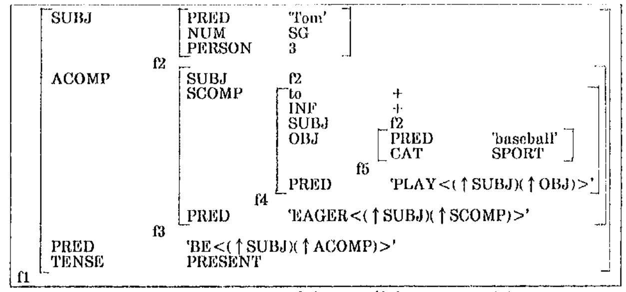

Fig. 4 F-structure of the English sentence (1)

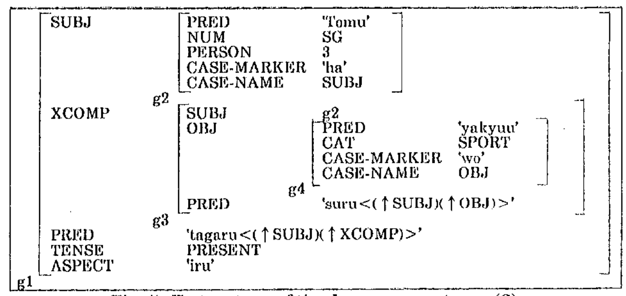

Fig. 5 F-structure of the Japanese sentence (2)

(a)(f_1 PRED) = 'BE <...>' (d)(f_4 PRED) = 'PLAY<...>'
(b)(f_2 PRED) = 'Tom' (e)(f_5 PRED) = 'baseball'
(c)(f_3 PRED) = 'EAGER <...>'

'be': Look up 'be' ; (f_1 PRED) = 'BE<...>'. The designator in the dictionary (table 1) is '$\uparrow = \downarrow$'. So $\uparrow = \downarrow$ = 'f_1'. Select the rule (2 a, b, c, d) in table 1.

(2) Check the conditions . Assign actual variable f_1 to the metavariable $\uparrow$. Unify the schemata of conditions with the f-structure (Figure 4). Then actual variables 'f_2' and 'f_4' are assigned to the metavariables $\downarrow$ and the following f-descriptions are obtained.

E [(f_1 SUBJ) = f_2] E [(f_1 ACOMP SCOMP) = f_4]

All the conditions of the (rule 2) are satisfied. Write 'f_2' and 'f_4' in the table 5 in Figure 4.

(3) Instantiate the schemata in the result side. For rule (2.a), look up in the table 5. There is no actual variable corresponding to 'f_1'. So assign a new actual variable 'g_1' to the metavariable $\uparrow$. Write actual variable 'g_1' corresponding to 'f_1' in the table 5 .

E [(f_1 SUBJ) = f_2] <====> J [(g_1 PRED) = g_2] ...(1)

Likewise,we get the other f-descriptions (2) (3) from rule (2.a), the f-descriptions (4), (5) from rule (2.b), the f-descriptions (6) from rule (2.c) and the f-descriptions (7), (8) from rule (2.d).

'Tom': the f-descriptions (9), (10), (11) are obtained.
'eager' : the same f-descriptions (1)...(8) are obtained.
'play' : the f-descriptions (12)..(18) are obtained.
'baseball' : the f-descriptions (19),(20) are obtained.

(4) Solve the f-descriptions (1)...(20) below. So the target f-structure (Figure 5) is obtained.

Table 6

(1)J [(g1 SUBJ) = g2]	(11)J [(g2 PERSON) = 3]
(2)J [(g1 SUBJ CASE-MARKER) = 'ha']	(12)J [(g3 SUBJ) = g2]
(3)J [(g1 SUBJ CASE-NAME) = SUBJ]	(13)J [(g3 SUBJ CASE-MARKER)='ha']
(4)J [(g1 PRED) = 'tagaru<...>']	(14)J [(g3 SUBJ CASE-NAME) = SUBJ]
(5)J [(g1 XCOMP SUBJ) = (g1 SUBJ)]	(15)J [(g3 OBJ) = g4]
(6)J [(g1 XCOMP) = g3]	(16)J [(g3 OBJ CASE-MARKER) = 'wo']
(7)J [(g1 TENSE) = PRESENT]	(17)J [(g3 OBJ CASE-NAME) = OBJ]
(8)J [(g1 ASPECT) = 'iru']	(18)J [(g3 PRED) = 'suru <...>']
(9)J [(g2 PRED) = 'tomu']	(19)J [(g4 PRED) = 'yakyuu']
(10)J [(g2 NUM) = SG]	(20)J [(g4 CAT) = sport]

Acknowledgment

The author would like to thank Prof. Narita of Waseda University, Mr. Ookawa, chairman of CRI (CSK Research Institute) and Mr.Yada, president of CRI for their constant encouragement.

References

[1].R.M.Kaplan and J.Bresnan, 'The Mental of Grammatical Relations', MIT Press, 1982.