The General System Design of KSHALT

Kwangseob Shim and Yung Taek Kim

This paper presents the general system design of an English-Korean machine translation system (KSHALT). The system has been developed as a prototype for practical machine translation systems and is based on the transfer approach. It consists of three components: an English syntactic parser, an English-Korean transfer and a Korean morphological generator. PEG (PLNLP English Grammar) is used as an English parser. Other components are witten is Common Lisp.

1. Introduction

KSHALT is an English-Korean machine translation system based on the transfer approach. In the logical point of view, it consists of three components: an English syntactic parser, an English-Korean transfer and a Korean morphological generator. Figure 1 shows the logical structure of KSHALT in detail. As the figure shows, English syntactic analysis is performed by PEG (PLNLP English Grammar). Karen Jensen gave detailed description of PEG in Jensen (1986). Figure 2 shows an example of parse trees produced by PEG. The parse trees are modified into augmented phrase structure trees by the interface component. Figure 3 shows a phrase stucture tree modified by the component. AVM (Attribute Value Matrix) annotated to each node is omitted in the figure because of limited space.

The English-Korean transfer component consists of three stages: PRE-TRANSFER, LEXICAL TRANSFER and POST-TRANSFER. The selection of target language equivalents is done at LEXICAL TRANSFER stage. Structural transfer (transformation) is done at both PRE and POST-TRANSFER stages. The former performs lexically independent transformation while the latter does lexically dependent transformation.

We made an attempt to give more importance to the LEXICAL TRANSFER stage than to the PRE and POST-TRANSFER stages. This means that the bilingual lexicon plays an important role through the translation

process. In this approach, the bilingual lexicon is supposed to provide most of the bilingual lexical information that is necessary for the translation process.

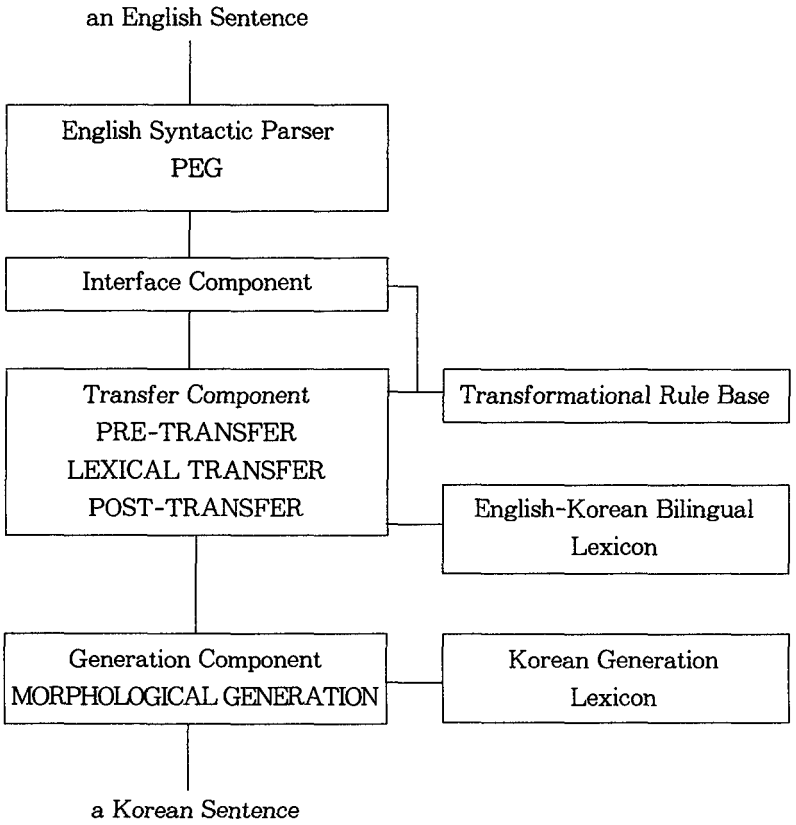


Figure 1. The Logical Structure of KSHALT

Therefore, the Korean generator becomes trivial. Only morphological synthesis is done by the generation component. The Korean generation lexicon is merely a set of morphological synthesis rules.

Though the current bilingual lexicon has been built for the translation of computer manuals, the system is not dedicated only to the translation of them. Changing the domain-dependent bilingual lexical information and adding new terminologies to the lexicon will be enough for the system to be

applied to other domains than computer manuals.

DECL	NP	PRON* "it"	
	VERB*	"consists"	
	PP	PP	PREP* "of"
		QUANTP	ADJ* "three"
		NOUN*	"components"
	PUNC	"."	

Figure 2. A Parse Tree Produced by PEG

```
(SENT (NP    (NOUN  "it"))
      (VP    (VERB  "consist")
             (PP    (PREP "of")
                    (NP  (QUANTP  (ADJ "three"))
                         (NOUN    "component")))))
```

Figure 3. A Modified Parse Tree

We performed an experiment with the system and translated an IBM manual 'SQL/Data System Concepts and Facilities' written in English. The translation quality of the system was evaluated using a grading scheme based on easiness of post-editing. According to the evaluation, about 60 percent of the manual turned out to be *class A*. Those sentences of class A can be used without post-editing.

2. Physical Structure of KSHALT

The physical structure of KSHALT is different from the logical one. Only two kinds of linguistic information are used by the system: the (transformational) rule base and the (bilingual and generation) lexcon. Therefore, KSHALT consists of two subsystem; *Transformation subsystem* to interpret the transformational rule base and *lexicon subsystem* to interpret the information stored in a (bilingual or generation) lexicon. The physical structure

of KSHALT is shown in Figure 4. Although PEG is used in KSHALT, it is not developed by the authors and thus it is not included in this figure.

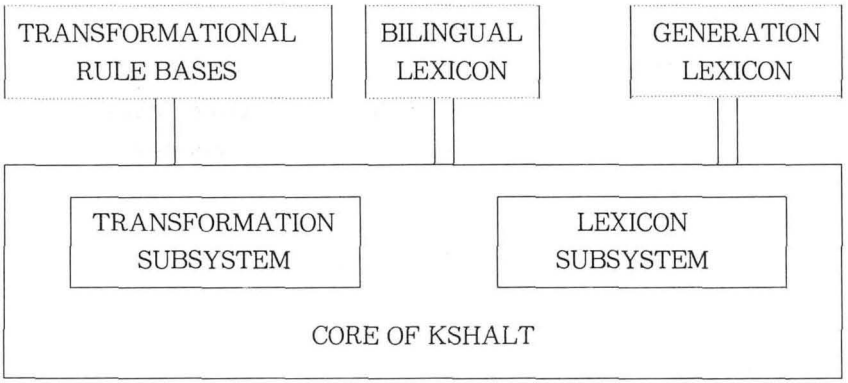


Figure 4. The Physical Structure of KSHALT

As shown in the figure, the transformational subsystem and lexicon subsystem form the core of KSHALT. This is fixed part of KSHALT. To the contrary, the transformational rule base, bilingual lexicon and generation lexicon are variable part of KSHALT. They could be expanded in size, if necessary. This is an easy way of building such a sophisticated system as a machine translation system.

3. Transformational Rule Base

The transformational rule base consists of a set of transformational rule groups. A series of transformational rules forms a transformational rule group. Transformational rule groups are applied one by one as specified in a RASS file. An example of the RASS file is shown in Figure 5. This file defines the logical structure of KSHALT. The first line is a header of the file. The second line corresponds to the interface component and the third line corresponds to the PRE-TRANSFER stage and so on. In the RASS file, each line has two fields. Transformational rule group(s) in the first field will be applied in a top-down mode while transformational rule group (s) in the second field will be applied in a bottom-up mode.

```
(RULE APPLICATION SEQUENCE SPECIFICATION)
  (INTERFACE 1)          (INTERFACE 2)
  (PRETRAN 1)            (PRETRAN 2)
  ()                     (LEXTRAN 1)
  (POSTRAN 1)            (POSTRAN 2)
  (MORPHGEN 1)           (MORPHGEN 2)
```

Figure 5. An Example of RASS File

Each transformational rule is used to transform one tree structure into another tree structure. Basically a transformational rule consists of a pair of a matching pattern and a target pattern. Lisp functions can also be employed in a transformational rule. If a parse tree (or a subtree) matches the matching pattern of a transformational rule, then the tree (or a subtree) is transformed to another tree according to the target pattern of the rule. An instance of transformational rules is shown in Figure 6. The leading dollar sign designates a matching variable that could be bound to any data type. Rule number is used as an identifier when the tracing or stepping facilities are active. If these two rules are applied to the parse tree shown in Figure 2, then the tree will be transformed to the new tree as shown in Figure 3.

```
(1001)   (DECL (NP $1) (VERB* $2) $3)
         → (SENT (NP $1) (VP (VERB $2) $3))
(1002)   (PP (PP (PREP* $1)) $2)
         → (PP (PREP $1) (NP $2))
```

Figure 6. An Example of Transformational Rules

The first element of a matching pattern designates candidate phrases where the rule can be applied. For example, the rule (1002) will be applied only to a prepositional phrase (PP) if the phrase matches the matching pattern of the rule. As a matter of fact, there would be more than one rule that could be applied to a single phrase. In this case, the most specific rule is applied. Here, the most specific rule means a rule which has the most specific matching pattern.

4. Bilingual Lexicon

At the stage of lexical transfer, a parse tree of an augmented phrase structure tree is traversed in a predetermined order. A phrase that is currently visited is called a current node. When a head word is a daughter of the current node, bilingual lexical information for the word is retrieved from the bilingual lexicon and applied to the current node. We assume that the bilingual lexicon should provide most of the bilingual lexical information for the translation process. For the description of bilingual lexical information we developed a dictionary description language DDL/I (Shim & Kim (1991)). Conceptually it can be divided into two part: lexicon description primitives and macros. A brief description of DDL/I will be given in the following sections.

4.1. Lexicon Description Primitives

Some of the lexicon description primitives are listed in Figure 7. Boldfaced letters denote the names of the primitives and italic letters with angled brackets denote the arguments of the primitives. Each primitive is a function that returns ture or false.

(**daughter-is** <*node*> <*control*> <*ddl/i statement list*>)
(**mother-is** <*node*> <*ddl/i statement list*>)
(**ancestor-is** <*node*> <*ddl/i statement list*>)
(**precede** <*ddl/i statement list*>)
(**preceded-by** <*ddl/i statement list*>)
(**test-attribute** <*attribute*> <*value list*>)
(**set-attribute** <*attribute*> <*value*>)
(**bind** <*var*>)
(**insert** <*phrase*>)
(**substitute** <*var*> <*var*>)
(**delete**)

Figure 7. Lexicon Description Primitives

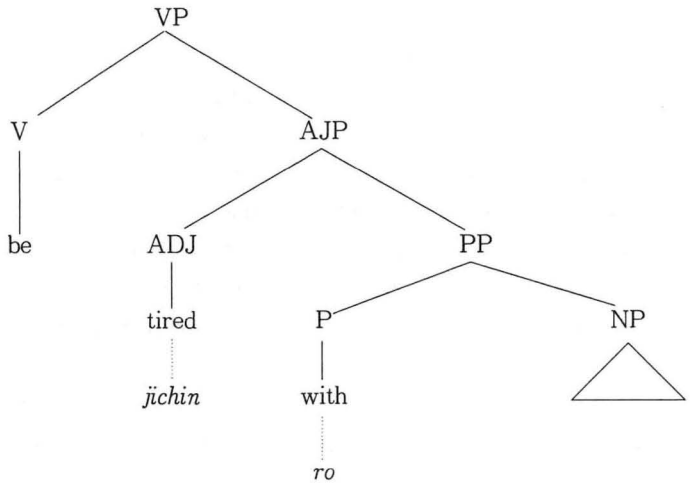
Assume that N_j is the current node. **daughter-is** sets N_k to be a new cur-

rent node in case N_k is immediately dominated by N_j and is equal to the <*node*>. According to the <*control*> the primitive will show different behaviors. **mother-is** sets N_i to be a new current node in case N_i immediately dominates N_j and is equal to the <*node*>. **ancestor-is** sets N_h to be a new current node in case N_h dominates N_j and is equal to the <*node*>. **precede** sets N_p to be a new current node in case N_p immediately precedes N_j **preceded-by** sets N_f to be a new current node in case N_f immediately follows N_j. For all these primitives, <*ddl/i statement list*> will be executed with the new current node and then the current node will be restored to N_j. **test-attribute** tests if the value of <*attribute*> of the current node is equal to one of the <*value list*>. **set-attribute** sets the value of <*attribute*> of the current node to the <*value*>. This function always returns true. **bind** assigns the current node to the <*var*>. **insert, substitute** and **delete** are used for the transformation of trees. These functions always return true as well.

Using these primitives, lexicographers or individual users can build a bilingual lexicon. Assume, for instance, a phrase structure tree for VP shown in Figure 8(a). For the present, ignore the Korean equivalents attached to the tree. They will be attached after the lexical transfer stage. AVM is not shown in this figure for simplicity. We can encode bilingual lexical information for an adjective "*tired*" into the English-Korean bilingual lexicon as shown in Figure 8(b).

Assume that AJP is the current node. As mentioned above, bilingual lexical information for the head word will be retrieved from the bilingual lexicon. In this case, bilingual lexical information for the adjective "*tired*" will be retrieved and it will be applied to the current node. **daughter-is** temporarily changes the current node to a prepositional phrase (PP) and tests if the head word of the PP is equal to "*with*". If it proves true, a Korean equivalent "*ro*" is attached there. Now, AJP is restored as the current node and a Korean equivalent "*jichin*" is attached there. The result is shown is Figure 8 (a).

(a)



(b) (ADJ "tired"
((**daughter-is** PP 1
(**test-attribute** HEAD ("with"))
(**set-attribute** TARGET "*ro*"))
(**set-attribute** TARGET "*jichin*")))

Figure 8. An Entry of a Bilingual Lexicon

Although the primitives are very flexible and powerful, they are not friendly to the lexicographers. Moreover, a bilingual lexicon which is encoded using lexicon description primitives will become complicated in structure and big in size. Macro definition facility has been adopted to give user friendliness and to make a bilingual lexicon look smart.

4.2. Macro Definitions

The lexicon description primitives provide basic functions for the lexical transfer stage. However, it seems difficult for lexicographers to create a bilingual lexicon using them. Macro definition facility has been adopted in order to provide a convenient and extremely flexible way of expression. Figure 9 shows some macros defined on the lexicon description primitives.

```
(defmacro H (&rest word)
            (test-attribute HEAD word))
(defmacro T (word)
            (set-attribute TARGET word))
(defmacro ↓NP (&rest ddl/i-statement-list)
            (daughter-is NP 1 ddl/i-statement-list))
(defmacro ↓PP (&rest ddl/i-statement-list)
            (daugher-is  PP 1 dd/i-statement-list))
(defmacro POST (postposition)
            (set-attribute POST postposition))
```

Figure 9. An Example of Macro Definitions

With the macros we can rewrite bilingual lexical information for an adjective "*tired*" as follows. It is simpler than that shown in Figure 8(b).

```
(ADJ "tired"
     ((↓PP (H "with") (T "ro")) (T "jichin")))
```

Natural language is so complicated that unexpected problems might occur while developing a machine translation system. In this case, our approach seems very flexible and powerful. In addition, our approach is not language-dependent, so it could be applied to any pair of languages with proper definitions of macros.

4.3. Bilingual Lexicon

With the lexicon description primitives and macros defined in the previous section, we will give an example of an English-Korean bilingual lexicon.

(1) This chapter deals with the following.
(2) An exception code informs the application of the reason for the failure.

Bilingual lexical information for the verbs "deal" and "inform" can be described as follows.

```
(VERB"deal"
     ((↓PP (H "with") (T "eul"))
```

(T "*taru-da*")))
(VERB "inform"
((↓NP (POST "*eke*")) (↓PP (H "of") (T "*eul*")) (T "*alli-da*")))

The following sentences have idiomatic expressions that are extracted from Schenk (1986). Note that underlined words form idioms.

(3) Pete kicked the bucket.
(4) He had made his peace with his neighbours.

Bilingual lexical information for idioms is not a special case while idioms are considered specially in many other systems. We can describe bilingual lexical information for idiomatic expressions as we did above for non-idiomatic expressions.

(VERB "kick"
((↓NP (H "bucket") (**delete**)) (T "*zuk-da*")))
(VERB "make"
((↓NP (H "peace") (**delete**)) (↓PP (H "with") (T "*wa*")) (T "*hwahaeha-da*")))

In this section we have shown several examples only for verbs. Bilingual lexical information for other lexical categories can be described in a similar way.

5. Experiment

KSHALT has been developed as a prototype for practical machine translation systems. The English-Korean bilingual lexicon has about 25,000 entries. The Korean generation lexicon has about 150 entries. The transformational rule base has some 200 rules.

We made an experiment with the system. An IBM manual 'SQL/Data System Concepts and Facilities' written in English was translated by the system and a Korean version of the manual was published. The manual contains 2,649 sentences. The translation quality of the system was evaluated using grading scheme based on easiness of post-editing. The result of evaluation showed that about 60 percent of the sentences was *class A*. Those sentences of class A can be used without post-editing. Some of them

are shown in appendix. Figure 10 shows the criteria of evaluation and Figure 11 shows the result of evaluation.

Class	A	Excellent translation. No post editing is necessary.
Class	B	Post editing is possible without referring to the source sentence.
Class	C	Post editing is possible with referring to the source sentence.
Class	D	Unacceptable. Post editing is very difficult.
Class	F	Parsing errors.

Figure 10. Criteria of Evaluation

Class	Number of Sentences	Percent
A	1537	58%
B	363	14%
C	196	7%
D	139	5%
F	414	16%
TOTAL	2649	100%

Figure 11. The Result of Evaluation

6. Conclusion

KSHALT is a prototype machine translation system based on the transfer approach. As a prototype system the bilingual lexicon has been built for the translation of computer manuals. However, it is not dedicated only to the translation of them since the physical structure of KSHALT is independent of domains. Changing the domain-dependent bilingual lexical information and adding new terminologies to the lexicon will be enough for the system to be applied to other domains than computer manuals.

References

Alonso, Juan A. (1988) 'A Model for Transfer Control in the METAL MT-System,' *Proc. of COLING '88*, pp. 19-23.

Golan, Igal, Shalom Lappin and Mori Rimon(1988) 'An Active Bilingual Lexicon for Machine Translation,' *Proc. of COLING '88*, pp. 205-211.

Jensen, Karen (1986) 'PEG 1986: A Broad-Coverage Computational Syntax of English,' *Technical Report*, IBM T. J. Watson Research Center.

McCord, Michael C. (1989) 'Design of LMT: A Prolog-based Machine Translation System,' *Computational Linguistics* 15, pp. 25-52.

Schenk, Andre (1986) 'Indoms in the ROSETTA Machine Translation system,' *Proc. of COLING '86*, pp. 319-384.

Shim, Kwangseob and Kim, Yung Tack (1991) 'A Study on a Transfer Dictionary Description Language,' *Technical Report* TR-91-01-1, Seoul National University.

Steele, Guy L. JR. (1984) *Common LISP: The Language*, Digital Press.

Vauquois Bernado and Christian Boitet (1985) 'Automated Translation at Grenoble University', *Computational Linguistics* 11, pp. 28-36.

Appendix

With a CREATE VIEW command, a user defines and stores a view definition that refers to underlying tables.
CREATE VIEW 명령어를 사용하여 사용자는 기본 테이블을 참조하는 뷰의 정의를 정의하고 저장한다.

The simplest type of a view is one that identifies a subset of one real table.
가장 단순한 타입의 뷰는 한 개의 실테이블의 부분집합을 식별하는 것이다.

Users possessing CONNECT authority can run ISQL and the DBS utility program.
CONNECT 권한을 가지고 있는 사용자는 ISQL과 DBS 유틸러티 프로그램을 실행할 수 있다.

Any user possessing RESOURCE authority can create tables in PUBLIC DBSPACE.
RESOURCE 권한을 가지고 있는 어떤 사용자라도 PUBLIC DBSPACE에

테이블을 생성할 수 있다.

A user with DBA authority can perform all operations on all tables, and can run all programs.
DBA권한을 가진 사용자는 모든 테이블에 대한 모든 조작을 수행할 수 있고 모든 프로그램을 실행할 수 있다.

It can modify the contents of SQL/DS catalog tables by an INSERT, DELETE, or UPDATE command.
그것은 INSERT, DELETE 또는 UPDATE명령어에 의해 SQL/DS카탈로그 테이블의 내용을 수정할 수 있다.

The accounting facility keeps track of the amount of processor time used by each user.
회계 기능은 각 사용자에 의해 사용된 처리기 시간의 양을 관리한다.

It also reflects each user's demand for the database and each user's demand for storage.
그것은 기억장소에 대한 각 사용자의 요구와 데이타베이스에 대한 각 사용자의 요구를 반영하기도 한다.

To use queries like this, the user must know the types of data in SYSCOLUMNS, and the names of the columns.
이와 같은 질의어를 사용하기 위하여 사용자는 SYSCOLUMNS에 있는 데이타의 타입과 그 열의 명칭을 알아야 한다.

The SQL/DS catalog and principal tables are summarized in Figure 26 on page 85.
SQL/DS카탈로그와 주요한 테이블은 85페이지의 그림 26에서 요약된다.

Department of Computer Engineering
Seoul National University
56-1 Shillim-dong Kwanak-gu
Seoul 151-742
Korea