A Prolog-based Korean-English Machine Translation System and Its Efficient Method of Dictionary Management

J.M. Choi, M.S. Song, K.J. Jeong, H.C. Kwon,
S.Y. Han, and Y.T. Kim

Department of Computer Engineering
Seoul National University
San 56-1, Shinrim-dong, Kwanak-ku
Seoul, Korea

ABSTRACT

This paper describes a Prolog-based Korean-English Machine Translation System (KEMTS).
KEMTS employs the transfer approach and consists of four separate phases - morphological analysis, parsing, deep structure generation, and English generation.
The implementation described here is based on a syntactic analysis of Hangul(the Korean language) and English, and it applies and extends the work of Marcus.
KEMTS also makes use of a dictionary management system called DProlog in order to overcome the inefficiency of the sequential search in Prolog and to manage the large amount of data conveniently.

INTRODUCTION

Prolog has shown some powerful aspects particularly in natural language processing because of its pattern matching and unification facilities (Clocksin and Mellish 1980), so KEMTS adopts it as a main implementation language.

KEMTS embodies the transfer approach (Tucker 1984), which separates the process of translation into three general stages : analysis, transfer, and synthesis. This approach allows for the separation of source language analysis from target language synthesis, parsing strategy from morphological analysis, and so forth.

Although based on the assumption that understanding is necessary for correct translation of text, the current system does not contain an understanding component.

KEMTS consists of four phases - morphological analysis, parsing, deep structure generation, and English generation. The first three of them are taking charge of the analysis stage from the general transfer approach, while the tasks of transfer and synthesis are performed in the last generation phase.

Deep structure is a language-free representation formalized via Chomsky-type transformational grammar (Chomsky 1957), and we focus analysis and synthesis around this medium.

Dictionary elements can be simply represented by predicates in Prolog, but as the amount of data increases, there occur some negative phenomena which lower the system performance. In order to solve this problem, we propose a dictionary management system named DProlog which supports

an efficient dictionary manipulation by storing dictionary elements in secondary storage, and by providing a fast search feature using a hashing mechanism.

DProlog also guarantees the abstraction that each phase of KEMTS can easily refer to the dictionary information by Prolog syntax, without the knowledge of the data format stored in secondary storage.

In the following sections, each phase of KEMTS will be explained, and some results obtained from experiments using this system will be reported on.

MORPHOLOGICAL ANALYSIS

Most English words are found in the dictionary exactly as written or in a form that follows from stripping off endings like -s, -ed, -ing.

Compared to English, Hangul morphology is complex because it strings together simple words to form compounds (Huh 1978; Lee 1970). For example, the word 'saramdolon' would not occur in the dictionary, but its components would. 'saram' means a human, 'dol' is a plural affix, and 'on' is a postposition which has a role of a subjective case marker. The case markers like this are very useful in analyzing Hangul sentences to form the correct parse structure.

In this phase, each Hangul word is classified into grammatically meaningful morphemes. This is considered to be unnecessary for English in which each word itself is one grammatical meaning.

The combination patterns of Hangul words are listed below (here [] indicates the optional case) :

1) noun
2) noun [+ plural affix] + postposition
3) adnominal (or adverb)
4) verb (or adjective) stem [+ aux. stem] + ending

Since matching each word with the dictionary elements is the main task in this analysis, Prolog's pattern match and backtracking facilities play an important role.

The outline of the morphological analysis procedure can be informally defined as follows :

< the morphological analysis procedure >

```
1.  WORD  <---  one input word

2.  match WORD with noun lexicon ;
    if matched then
       return with 'WORD is an isolated noun' ;

3.  match WORD with adnominal or adverb lexicon ;
    if matched then
       return with
       'WORD is an isolated adnominal or adverb' ;

4.  find a postposition P in the end of WORD ;
    if found then
          delete P from WORD ;
```

```
            match remaining part of WORD with noun lexicon ;
            return with
              'WORD is noun + postposition' ;

5.  make reverse list of WORD ;
    do next two tasks repeatedly and simultaneously
          until successful match occurs ;

       task1 :  scan WORD forward and match scanned string with
                verb or adjective lexicon ;
       task2 :  scan WORD backward using reversed list and match
                scanned string with word endings, or auxiliary
                stems representing tense or modal ;

    end
    if successfully matched then
       WORD is 'verb (or adjective) [+aux. stem] + ending' ;
       return ;

6.  error recovery routine
```

The only complicated part in the above procedure is the case of 'verb [+aux. stem] +ending'. We extract the string of each word by forward and backward scanning. In Prolog implementation backward scanning is enabled by the forward scanning of reversed list of input words.

PARSING

The aim of this and the next phase is to produce a parse tree of the sentence, together with sufficient semantic information to enable a complete and correct translation to be generated.

It is well-known that top down backtracking parsers are easy to write in Prolog, and those parsers can be extended to give them the power of ATN's (Pereira and Warren 1980). But this kind of top down parser has all the problems of standard context-free top down parsers ; left recursive rules can cause looping, and backtracking is quite inefficient.

In particular, Hangul has the syntactic constraints that a verb must come last and an embedded clause must come first in a sentence, which forces us to favor the bottom up parsing (Lee 1970; Stabler 1983).

Promising alternatives to context-free rule parsers and ATN parsers are the Wait-and-See Parsers (WASP) (Marcus 1980). WASP uses the deterministic bottom up strategy, and our parser is a Prolog implementation of WASP.

Our WASP consists of three components - a stack, a buffer, and a set of condition action rules. Sentences stream from right to left through the buffer, ending up in a standard looking parse tree.

The condition action rules control the flow of data between the stack and the buffer. The condition part of each rule is allowed to consider the contents of the buffer and the current node. The action part of each rule specifies one of the following actions.
(In these actions, 'OldSt' and 'NewSt' represent for the contents of the stack before and after the actions, respectively, while 'OldBuf' and 'NewBuf' for the contents of the buffer before and after the

actions, respectively) :

1) Create :
creates a new node(ToBeCreated) and pushes it onto the stack.

```
create(OldSt, ToBeCreated, NewSt) :-
    append(OldSt, ToBeCreated, NewSt).
```

2) Attach :
attaches the first item in the buffer(ToBeAttached) to the top of the stack, and shifts other buffer items left to fill the hole which was previously occupied by the removed item.

```
attach(OldSt,StLast,ToBeAttached,NewSt) :-
    append(StLast,ToBeAttached,St1),
    append(OldSt, [St1],NewSt).
```

3) Drop :
drops the most recently popped node(ToBeDropped) into the first position in the buffer, after shifting existing items to the right.

```
drop(OldSt,NewSt,ToBeDropped,OldBuf,NewBuf) :-
     NewSt = OldSt,
     append(ToBeDropped,OldBuf,NewBuf).
```

The condition action rules reside in a partitioned condition action rule memory. Each partition corresponds to a parse-tree node type. Only those rules in the partition associated with the current parse-tree node, the one at the top of the stack, are active.
One of these rules is as follows :

```
cond_act(OldSt,OldBuf,NewSt,NewBuf) :-
     stack_check(OldSt,Rem,LastElm,s),
     OldBuf = [[np|Temp]|NewBuf],
     create(OldSt, [[np|Temp]],NewSt).
```

The parsing procedure is quite simple because all the start, stop, and attach knowledge is locked up in the rules.
Let us look at the parsing procedure.

< parsing procedure >

```
until the sentence is completely parsed or
    no active condition action rule applies do
  1. activate the condition action rules
     associated with the current node
  2. use the first active rule that matches
     the condition of the buffer and the
     current node
end
```

Prolog implementation of this procedure is also simple and as follows :

```
parsing(Tree, [],Tree, []).
parsing(OldSt,OldBuf,NewSt,NewBuf) :-
    cond_act(OldSt,OldBuf,StTemp,BufTemp),
    parsing(StTemp,BufTemp,NewSt,NewBuf).
```

Our Prolog implementation of WASP can be extended to get great coverage of the language without the substantial increase in parsing time that would be expected for a strictly top down parser with some extensions.

GENERATION OF DEEP STRUCTURE

We define the deep structure as a hierarchical structure of clauses included in a sentence. This structure contains information about tense, mood and the meanings of postpositions, endings and auxiliaries (Chomsky 1957; Simmons 1972).

Due to the various features in Hangul, we should consider several problems peculiar to Hangul in this phase (Lee 1970).

First of all, it is necessary to construct the hierarchical structure of clauses in a sentence.
Second, we should analyze the meanings of postpositions, endings, and auxiliaries and add these to the deep structure.
Third, we should add information about tense and mood to the structure.

The information about tense and mood can easily be found in the morphological analysis phase because the auxiliary stems of mood and tense have a unique form corresponding to the meaning.
As a result of this, it is possible to solve the third problem easily by a simple movement mechanism (Chomsky 1957).

A postposition in Hangul is attached to an absolute word such as a noun or a pronoun, and determines the constituent of the absolute word in a sentence.

Generally postpositions are classified into three cases; subjective case, adnominal case, and adverbial case.
Postpositions of subjective case and adnominal case have fixed meanings, but in the adverbial case the meanings change according to sentences. Therefore, we have to decide the meaning of a postposition of adverbial case.

With priority given to a conjugational word such as a verb or an adjective, the meanings of postpositions with which are accompanied by a conjugational word are recorded in the conjugational word dictionary. At the same time, all meanings that each postposition can have are recorded in the postposition dictionary.
And then we compare the meanings of the postposition attached to a conjugational word with the meanings of the postpositions that the related conjugational word can accompany with. In this way we can determine the meanings of a postposition in a sentence.

In order to construct the hierarchical structure of clauses included in a sentence, we should understand the meanings of endings and analyze the relationships between included clauses.
We use the reverse transformations to solve these. The transformations are minutely explained in the literatures of transformational grammar (Chomsky 1957; Pereira and Warren 1980).

In Hangul, there are three kinds of clauses ; noun clauses, adnominal clauses, and adverbial clauses.

It is easy to find the embedded noun clause in a sentence. The nominal ending of noun clause has a unique form corresponding to its meaning,

so they can be checked during the morphological analysis phase.

The syntactic structure of a noun phrase modified by an adnominal phrase has the form of 'Adnp(S) + Np(N)', where 'Adnp(S)' is the adnominal phrase which modifies the noun phrase, 'Np(N)'. 'Adnp(S)' is to be transformed into an adnominal clause, the deep structure of 'Adnp(S)', which has all necessary sentential components.

To construct an adnominal clause of 'Adnp(S)', it is required to analyze the relationship between S and Np, and to add the information about the relationship.
We analyze the role of Np playing in S by applying a reverse transformation in Hangul.

In the embedded adverbial clause, each clause accompanies with the adverbial endings.
The method of making the deep structure of an adverbial clause is just to check the adverbial ending, to search the ending dictionary, and to record the meaning of ending in that dictionary.

ENGLISH GENERATION

Hangul differs from English not only in their phonological rules and lexicons, but also in their syntactic rules (Huh 1978; Lee 1970).

Considering this fact, English generation phase includes two processes, namely the substitution of words and the transformation of grammars.

Referencing dictionary elements is the main task in both processes, so we emphasize the clear construction of dictionaries for word and grammar information.

In words substitution, nouns are easily converted by one-to-one mappings of dictionary elements. But the handling of verbs is not the same because verbs of Hangul may be translated into idioms, not a single word of English.

For example, 'samda' of Hangul should be replaced by 'regard A as B', making the corresponding dictionary entry as follows :

```
v(sam, [regard,$obj,as,$adv]).
```

Here '$obj' will eventually be filled with the object constituent, and '$adv' with the adverbial constituent of the sentence.

The grammar transformation converts the word order of Hangul such as S+O+V into that of English such as S+V+O (Boyer and Lapalme 1985).
During this transformation, some exceptions can happen in which embedded clauses of Hangul had better be translated into English phrases, usually prepositional phrases.
For example, the English equivalent of a Hangul verb 'wihada' is not a verb of English but a preposition 'for'.

DICTIONARY MANAGEMENT SYSTEM : DPROLOG

KEMTS makes use of the dictionary management system called DProlog in

It deletes the clause 'pred_name(t1,t2,.....,tn)' from the dictionary.

(4) remove(pred_name).
It removes all relations named 'pred_name' from the dictionary.

(5) listall(pred_name).
It displays all clauses of the relation named 'pred_name'.

(6) ls.
It displays the information about all relations in the dictionary.

The current Prolog interpreter assumes that all ground unit clauses are stored in the main memory. Unfortunately the dictionary clauses which are similar to general ground unit clauses are not in the main memory. Therefore the mechanism which can distinguish the dictionary clauses on secondary device from the other clauses in main memory is required to retrieve the dictionary clauses.

Without any change of the Prolog interpreter, Dprolog distinguishes the dictionary clauses from the others using the communication predicate 'ext'.
The tasks which the communication predicate 'ext' accomplishes are listed below :

(1) 'ext' sends a query to the partial match system and receives the answer set from that system, via Unix pipe.

(2) 'ext' unifies the query and the first clause of the answer set which has not been unified yet.

(3) When a backtracking happens, 'ext' repeats the step 2.

An example is shown in Fig. 2. In this case, it is required only one rule 'srel(X,Y) :- ext(srel(X,Y))' to accomplish the communication in a dictionary procedure.

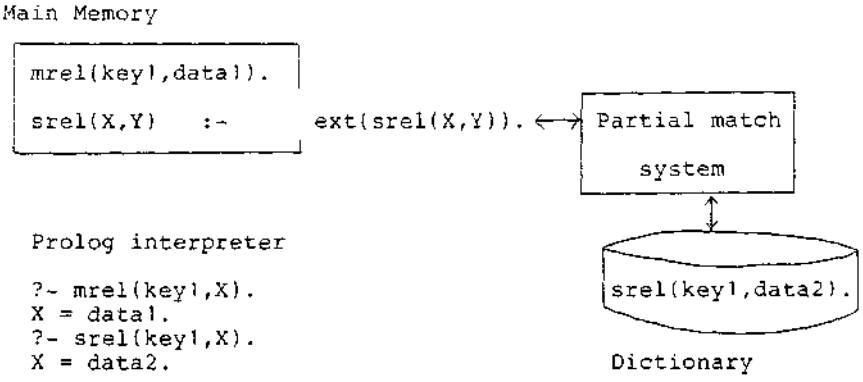

Fig. 2 The communication predicate 'ext'

DProlog stores each relation in a single file and retrieves the dictionary clauses in that file by partial match retrieval (Ramamohanarao and Lloyd and Thom 1983; Ullman 1982). Partial match retrieval is a method for efficient access to the files via multi-key combinations, and it is appropriate for Prolog.

RESULTS AND CONCLUSION

An experimental Prolog-based Korean-English Machine Translation System (KEMTS) has been introduced. In order to improve the overall performance, a dictionary management system called DProlog has also been presented.

KEMTS embodies the transfer approach that the translation is carried out by a syntactic and sentence-by-sentence manner. Another method of translation named 'the integral method' which uses active dictionary can be found elsewhere (Tanaka and Isahara 1983).

An example which shows the result of each phase is given in the Appendix.

KEMTS experienced somewhat satisfactory result with performance in the domain of Korean history, but severe problems may arise in a word selection when the ambiguity occurs. We have tried to solve this problem by consulting the remaining parts of the sentence, and this mechanism will be enhanced by understanding capabilities in the near future (Carbonell and Cullingford and Gershman 1981; Wilks 1973).

REFERENCES

Boyer M, Lapalme G (1985) Generating sentences from semantic networks. In: Dahl V, Saint-Dizier P (eds) Natural language processing and logic programming. North-Holland, 181-189

Carbonell J, Cullingford R, Gershman A (1981) Steps toward knowledge-based machine translation. IEEE Trans on Pattern Analysis and Machine Intelligence 3: 376-392

Chomicki J, Grudzinski W (1983) A database support system for Prolog. Proc Logic programming workshop, Algarve Portugal

Chomsky N (1957) Syntactic structures. Mouton, The Hague

Clocksin W, Mellish C (1980) The Unix Prolog system. Software Report 5, Univ of Edinburgh

Huh W (1978) Korean phonology, revised edn. Cheung-Eum Sa, Seoul Korea

Lee H (1970) A study of Korean syntax. Pan Korean Book Corporation, Seoul Korea

Marcus M (1980) A theory of syntactic recognition for natural language. MIT Press, Cambridge MA

Minker J, Tran D (1981) Interfacing predicate logic languages and relational databases. TR 1019, Univ of Maryland

Naish L, Thom J (1983) The MU-Prolog deductive database. TR 83 [illegible], Univ of Melbourne

Pereira F, Warren D (1980) Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks. Artificial Intelligence 13: 231-278

Ramamohanarao K, Lloyd J, Thom J (1983) Partial match retrieval using hashing descriptors. ACM Trans on Database Systems 8: 552-576

Simmons R (1972) Semantic networks : their computation and use for understanding English sentences. Tech Report NL-6, Univ of Texas at Austin

Stabler E (1983) Deterministic and bottom-up parsing in Prolog. Proc Nat Conf on AI, Amer Assoc for AI, 383-386

Tanaka H, Isahara H (1983) An English-Japanese machine translation system using the active dictionary. New Generation Computing 1: 179-185
Tucker A (1984) A perspective on machine translation : theory and practice. Comm ACM 27: 322-329
Ullman J (1982) Principles of database systems, 2nd edn. Computer Science Press, Rockville Maryland
Wilks Y (1973) An artificial intelligence approach to machine translation. In: Schank R, Colby K (eds) Computer models of thought and language. W.H. Freeman and Company, San Francisco

APPENDIX

An example which shows the result of each phase is given below.

Input sentence in Hangul ===>

조선은 국초부터 농본 민생의 기본 정책을 내세워 농업을 천하의 대본으로 삼았다.

The output of morphological analysis ===>

[nn(조선,0), jo(은,1), nn(국초,0), jo(부터,3), nn(농본,0), nn(민생,0), jo(의,2), nn(기본,0), nn(정책,0), jo(을,4), v(내세우,0), ee(어,31), nn(농업,0), jo(을,4), nn(천하,0), jo(의,2), nn(대본,0), jo(으로,3), v(삼,0), vi(았,2), ee(다,11)].

The output of parsing ===>

[[s, [advp, [s, [sjp, [np, [n,조선]],0], [predp, [advp, [np, [n,국초]], [post,부터]], [objp, [np, [adnp, [np, [cn, [n,농본], [n,민생]]]], [np, [cn, [n,기본], [n,정책]]]]], [pred, [v,내세우], [tns, 1]]], [dcl,1]], [aux,어]], [predp, [objp, [np, [n,농업]]], [advp, [np, [adnp, [np, [n,천하 [[[, [np, [n,대본]]], [post,으로]], [pred, [v,삼], [tns,2]]], [dcl,1]]].

The deep structure ===>

[[s, [ss, [advp, [ss, [sjp, [np, [n,조선]],0], [predp, [advp, [np, [n,국초]], [post,부터 ,6]], [objp, [np, [adnp, [np, [cn, [n,농본], [n,민생]]]], [np, [cn, [n,기본], [n,정책]]]]], [pred, [v,내세우], [tns,2]]], [dcl,1]], [aux,3,어]], [predp, [objp, [np, [n,농업]]], [advp, [np, [adnp, [np, [n,천하]]], [np, [n,대본]]], [post,으로 ,10]], [pred, [v,삼], [tns, 2]]], [dcl, 1]]].

Generated English sentence ===>

From the beginning of the nation, Choseon hung out the basic policy of physiocracy and public welfare, and regarded the agriculture as the world basic.