An Efficient Execution Method for Rule-Based Machine Translation

Hiroyuki KAJI

Systems Development Laboratory, Hitachi Ltd.
1099 Ohzenji, Asao, Kawasaki, 215, Japan

ABSTRACT

A rule based system is an effective way to implement a machine translation system because of its extensibility and maintainability. However, it is disadvantageous in processing efficiency. In a rule based machine translation system, the grammar consists of a lot of rewriting rules. While the translation is carried out by repeating pattern matching and transformation of graph structures, most rules fail in pattern matching. It is to be desired that pattern matching of the unfruitful rules should be avoided. This paper proposes a method to restrict the rule application by activating rules dynamically. The logical relationship among rules are pre-analyzed and a set of antecedent actions, which are prerequisite for the condition of the rule being satisfied, is determined for each rule. In execution time, a rule is activated only when one of the antecedent actions are carried out. The probability of a rule being activated is reduced to near the occurrence probability of its relevant linguistic phenomenon. As most rules relate to linguistic phenomena that rarely occur, the processing efficiency is drastically improved.

1. Introduction

A practical machine translation system needs to deal with a wide variety of linguistic phenomena. A large and sophisticated grammar will be developed over a long period. Accordingly, it is necessary to adopt an implementation method which improves the extensibility and maintainability of the system. The rule based approach [1] is a promising one from this viewpoint.

However, a rule based system is generally disadvantageous in processing efficiency. In rule based machine translation, a grammar is comprised with a lot of rewriting rules [2][3][4]. Translation is carried out by repeating pattern matching and transformation of tree or graph structures that represent the syntax or semantics of a sentence. A great part of the processing time is spent in pattern matching, which mostly results in failure. The key to improve the processing efficiency is how to avoid the pattern matching that results in failure.

A number of methods such as the Rete pattern match algorithm [5] have been developed to improve the processing efficiency of rule based systems. However, peculiarities in machine translation systems make it difficult to apply the whole of an existing method. The general idea of existing methods is to restructure the set of rules in a network such as a cause-effect graph, or a descriminant network, and maintain the state of the object in the network. The following are distinguishing features of a machine translation system. First, the object data is a graph structure, and the state of the object must be handled as a collection of states of respective subgraphs, which are created dynamically by applying rules. Therefore, maintaining the state of the object in a network causes a large amount of overhead. Secondly, rules are applied in a controlled manner, so that a linguistically insignificant result is prevented. The computational control of rules to improve the processing efficiency must be superimposed on the linguistic control of rules.

This paper proposes a new method to improve the processing efficiency of rule based systems having the above mentioned features. Section 2 describes a grammar description language which was developed for a Japanese-English machine translation system. Though the proposed method is described on the basis of this grammar description language, it is general enough to apply to other systems. Section 3 explains the problem of processing efficiency. Then, Section 4 outlines the proposed method by which essence is in dynamic rule activation, based on the logical relationship among rules. A method to pre-analyze the logical relationship among rules is described. The improved grammar executor is also described. Lastly, the effectiveness of the proposed method is discussed in Section 5.

2. Grammar Description Language for Rule Based Machine Translation

2.1 Object data structure

A machine translation system deals with the syntax and semantics of a natural language sentence, which is represented by tree or graph structures. The object data in our machine translation system is a directed graph. A directed graph consists of a set of nodes and arcs connecting a pair of nodes. Each node has a number of attributes and each arc has a label. The label of an arc can be regarded as a kind of attribute in the tail node of the arc. The attributes are divided into scalar-type attributes and set-type attributes. A scalar-type attribute is one in which only one value is given to a node. A set-type attribute is one in which more than one value may be given to a node.

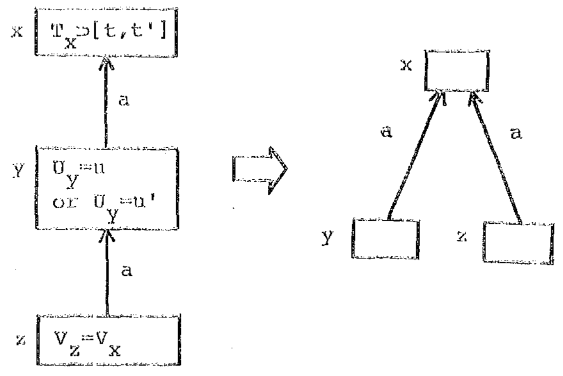
In Japanese-English machine translation, a node corresponds to a bunsetsu in a Japanese sentence. A bunsetsu is comprised with a content word and the succeeding function words. The following are treated as attributes of nodes; parts of speech, semantic features, function words, dependent types, governor types, surface case markers, semantic roles (case), and others.

2.2 Grammatical rules

A grammatical rule is written in the form of a graph-to-graph rewriting rule. That is, a rule consists of a condition part and an action part. The condition part specifies the pattern of a

```
condition
* @X : T ⊃ [ t, t' ]
     ( a : @Y ) ;
     @Y : U = u ! u'
        ( a : @Z ) ;
        @Z : V = @X.V ;
action
  @X ( + a : @Z ) ;
     @Y ( - a : @Z ) ;
```

(a) Coding form



(b) Illustrative form

Fig. 1 An example of a grammatical rule

subgraph, and the action part does a transformation to be performed on subgraphs that match the pattern specified in the condition part. Fig. 1 shows an example of rule. In Fig. 1, (a) is the coding form and (b) is an illustrative form. As nodes are represented by variables (character strings headed by @), rules should be applicable to any subgraph in the object data. A rule has a key node variable, which is indicated by *. The key node plays a role in specifying exactly the location where the rule is applied in the object data.

The condition part of a rule is a logical combination of primitive conditions. A primitive condition is related to either a node connection or an attribute. Equality is specified for a scalar-type attribute, and an inclusion relationship is specified for a set-type attribute. The primitive conditions are also divided into intra-node conditions and inter-node conditions.
- An intra-node condition is one relating to only one node.
 e.g., @X : T ⊃ [t, t'] ;
 The set-type attribute T of node @X includes the values t and t'.
- An inter-node condition is one relating to a pair of nodes.
 e.g., @X : T = @Y.T ;
 The attribute T of node @X has the same value as that of node @Y.

The action part of a rule is a sequence of primitive actions. A primitive action is related to either a node connection or an attribute. Connection and disconnection are specified for a pair of nodes. Substitution of a value is specified for a scalar-type attribute, and addition and deletion of a value are specified for a set-type attribute. The actions are also divided into intra-node actions and inter-node actions.
- An intra-node action is one relating to only one node.
 e.g., @X : T = T + [t] ;
 Add a value t to the set-type attribute T of node @X.
- An inter-node action is one relating to a pair of nodes.
 e.g., @X : T = @Y.T ;
 Substitute the value of attribute T of node @Y for the attribute T of node @X.

2.3 Application control of rules

A grammar consists of a lot of rules, which play their own roles in the translation process. They must be applied in a controlled manner, so that linguistically insignificant results are prevented. The grammar description language provides a facility to modularize a grammar and specify sophisticated control in rule application.

A grammar is decomposed into a lot of subgrammars, which are applied in a prescribed order. For example, the analysis grammar for Japanese sentences is decomposed into such subgrammars as disambiguation of multiple parts of speech, determination of governor types, determination of dependent types, dependency structure analysis, deep case analysis, tense/aspect analysis, and others. A subgrammar may be decomposed into further subgrammars.

A number of control parameters for rule application are specified for each subgrammar. The following are examples.
- Mutual relationship among rules (Exclusive, Concurrent, Dependent or Unrelated): For instance, when Exclusive is selected, rule application is controlled so that successful application of a rule should prevent the remaining rules from being applied.
- Traverse mode in the object data (Pre-order or Post-order): The object data is traversed in the specified mode, and rules are applied at each location in the object data structure.
- Priority between rule selection and location selection: When rule selection is selected, rule application is controlled so that the next rule should be selected after applying a rule at every location.

3. Problem of Processing Efficiency

A naive implementation of grammar executor for such a grammar description language as described in Section 2 is illustrated in Fig. 2. The translation is carried out by applying grammatical rules to the object data in the working memory. The grammar executor consists of the initializer, the controller, the pattern matcher and the transformer.

The initializer creates an initial state of the object data in the working memory, based on the result of morphological analysis. It defines a node for each bunsetsu and assigns it some attribute values. The attribute values come from the dictionary and the result of morphological analysis.

The controller is initiated after the initial object data is created. The controller determines both the rule to be applied and the current node at which the rule is to be applied, according to rule application control parameters and the application result of the previous rule.

The pattern matcher judges whether the condition part of a rule is satisfied or not. The rule and the current node is designated by the controller.

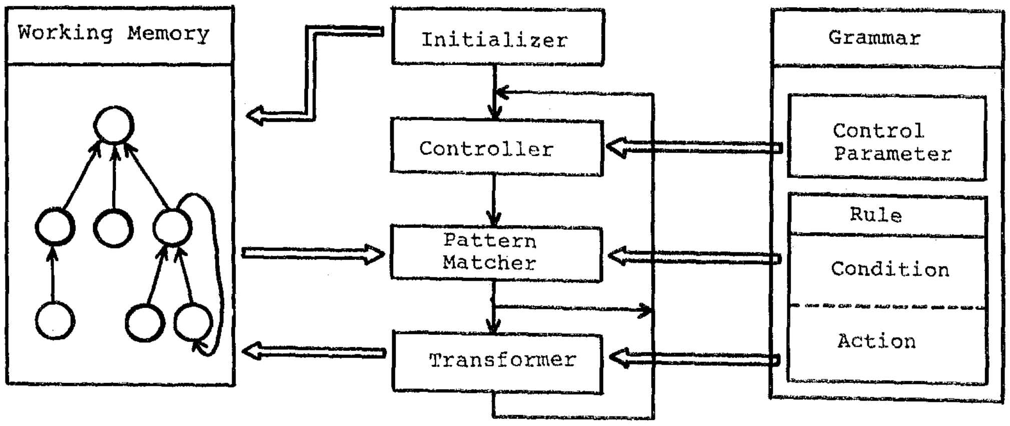


Fig. 2 Grammar executor

The pattern matcher first binds the key node variable in the rule with the current node. Then, it binds the other node variables with nodes in the object data one after another, searching for a node which satisfies the conditions relevant to each node variable. If all the node variables in the rule are bound with nodes, the pattern matcher judges that the condition part of the rule is satisfied at the current node. If there exists a node variable that cannot be bound with a node, the pattern matcher judges that the condition is not satisfied at the current node.

The transformer performs the action part of a rule. It is called only when the pattern matcher judges that the condition part of the rule is satisfied. As the pattern matcher has bound each node variable with a node in the object data, the appropriate portion of the object data structure undergoes the transformation.

The grammar executor described above leaves room for improvement in efficiency. The behavior of rules in the naive grammar executor shows the following characteristics.
- The proportion of rules that succeed in pattern matching is very small. It is less than one percent in the case of our Japanese sentence analysis grammar which is comprised of several thousand rules.
- The probability that a rule succeeds in pattern matching varies widely with rules. While some rules succeed fairly frequently, most other rules rarely succeed.

In the naive implementation of grammar executor, all the rules are treated equally. As a result, a great part of the processing time is spent in pattern matching of unfruitful rules. If application of unfruitful rules can be avoided, the processing efficiency will be drastically improved. Some rules can be directly linked to specific words. Application of such word specific rules can be easily restricted by linking them with the dictionary. Our concern here is how to restrict application of general rules that cannot be linked directly to specific words.

4. Dynamic Rule Activation

4.1 Basic idea

Whether the condition part of a rule is satisfied or not generally depends on the results of preceding rules. The logical relationship among rules can be extracted by static analysis of the grammar. A considerable application of unfruitful rules will be prevented by using the logical relationship among rules.

First, we define an antecedent set for a condition. The antecedent set for a condition is such a set of actions as:
(i) carrying out a member action causes the possibility that the condition is satisfied, and
(ii) the condition is never satisfied if no member action is carried out.
Then, we define the inverse action for an antecedent set. The inverse action for an antecedent set is an action that cancels the effect of any member action of the antecedent set. An antecedent set and its inverse action can be used to dynamically change the status of a rule as follows. A rule is activated when a member action of the antecedent set for the condition of the rule is carried out. A rule is deactivated when the inverse action is carried out. It is obviously assured that a rule is active whenever its condition may be satisfied. Thus, the application of inactive rules can be skipped.

More than one antecedent set can usually be obtained for a condition. The optimal antecedent set is one that minimizes the probability of activating a rule. The optimal antecedent set is one of minimal antecedent sets. The minimal antecedent set is such an antecedent set as any subset is not an antecedent set for the same condition. In order to choose the optimal antecedent set among minimal antecedent sets, occurrence statistics of actions should be gathered using a corpus of text.

4.2 Pre-analysis of grammar

4.2.1 Antecedent set for primitive condition

We are not interested in all the antecedent sets but the optimal one for the condition of each rule. Therefore, we turn our attention to intra-node conditions. Intra-node conditions usually give us an effective antecedent set, while inter-node conditions do not.

The minimal antecedent sets for an intra-node condition are as follow. Here, antecedent sets are defined separately for each node (indicated by i below), as the truth value of a condition varies with nodes. It is necessary to consider two cases. One is that the attribute in the condition is not related to any inter-node action. The other is that the attribute in the condition is related to some inter-node actions.

(1) When the attribute is not related to any inter-node action, the truth value of a condition at a node i is effected only by actions at the same node i. Therefore, only the actions at the same node i are included in the antecedent set.
e.g., The minimal antecedent sets for a condition $T_i \supset [\ t,\ t'\ ]$ are $[\ T_i = T_i + [t]\ ]$ and $[\ T_i = T_i + [t']\ ]$.
A comment should be given on composite actions. For instance, $T_i = T_i + [\ t,\ t',\ t''\ ]$ is also an antecedent action. However, it is decomposed into $T_i = T_i + [\ t\ ]$, $T_i = T_i + [\ t'\ ]$ and $T_i = T_i + [\ t''\ ]$. Therefore, we exclude it from antecedent sets.
e.g., The minimal antecedent set for a condition $T_i \cap [\ t,\ t'\ ] \neq \phi$ is $[\ T_i = T_i + [t]\ ,\ T_i = T_i + [t']\ ]$.

(2) When the attribute is related to some inter-node actions, the truth value of a condition at a node i may be effected by actions at another node via an inter-node action (See Fig. 3). Therefore, the antecedent sets need to include the actions at all the nodes.
e.g., The minimal antecedent sets for a condition $T_i \supset [\ t,\ t'\ ]$ are $[\ T_j = T_j + [t]\ \mid\ j=1,\cdots,N\ ]$ and $[\ T_j = T_j + [t']\ \mid\ j=1,\cdots,N\ ]$.
e.g., The minimal antecedent set for a condition $T_i \cap [\ t,\ t'\ ] \neq \phi$ is $[\ T_j = T_j + [t]\ ,\ T_j = T_j + [t']\ \mid\ j=1,\cdots,N\ ]$.
In this case, obviously the antecedent sets for a rule are common to all the nodes.

On the other hand, we cannot obtain effective antecedent sets from an inter-node condition. For instance, the minimal antecedent set for an inter-node condition $T_i = T_j$ must include actions $T_i = T_i + [\ t\ ]$ (for any t), as $T_i = T_i + [\ t\ ]$ make true the condition together with $T_j = T_j + [\ t\ ]$. Accordingly, the minimal antecedent set includes a large number of actions and has a rather large occurrence probability.

4.2.2 Antecedent set for rule

A minimal antecedent set for a condition or a rule is synthesized by those for the constituent primitive conditions. For this purpose, the condition part of a rule is transformed into conjunctive canonical form. The conjunctive canonical form is a logical AND of terms, each term being a logical OR of one or more primitives. In Fig. 4, the condition part of the rule in Fig. 1 is shown in conjunctive canonical form.

In the conjunctive canonical form, a term is true if anyone of the primitives is true, and it is false if all the primitives are false. Therefore, the union of the minimal antecedent sets of the primitives is that for the term. Here, the detailed procedure is separated into two cases. In the case of the term being related to the key node variable in the rule, the minimal antecedent sets for the node concerned should be united. On the contrary, in case the term is related to a node variable other than the key node variable, the minimal antecedent sets for all the nodes should be united, because any node may, as a result of structural change, occupy the location that corresponds to the node variable the term is related to (See Fig. 5).

The condition, a logical AND of terms, is totally true if and only if all the terms are true. Accordingly, each minimal antecedent set for one of

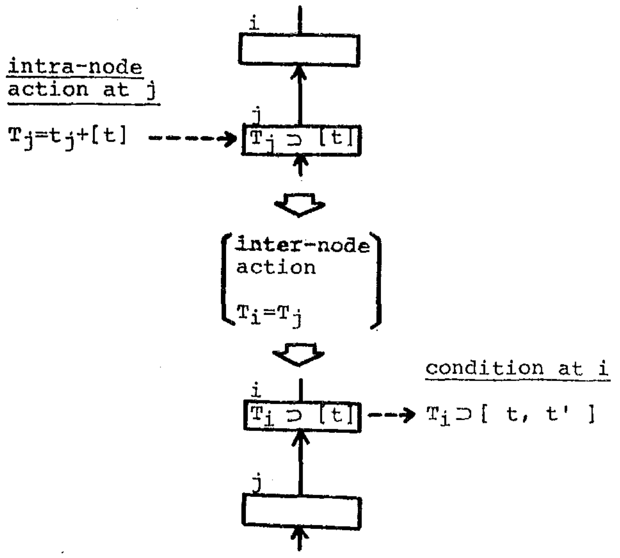


Fig. 3 Antecedent action via inter-node action

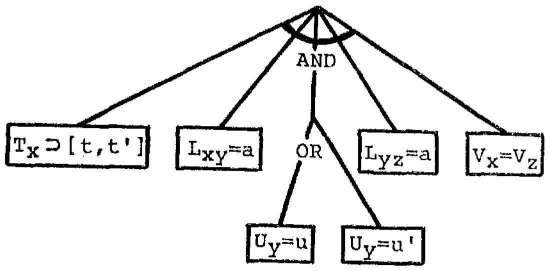


Fig. 4 Decomposition of a condition

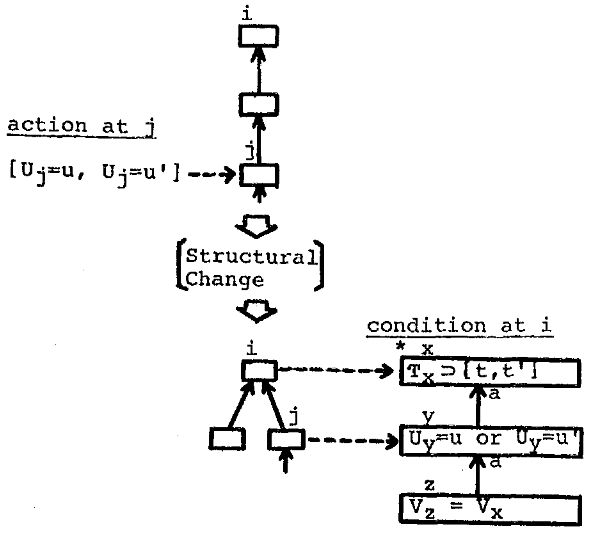


Fig. 5 Antecedent set via structural change

the terms is that for the condition. As the condition part of a rule usually includes one or more terms comprised of intra-node conditions, it does not matter that effective antecedent sets cannot be obtained from inter-node conditions.

As an example of the minimal antecedent set for a rule, those for the rule in Fig. 1 are given below.

[$T_i = T_i + [t]$] ,
[$T_i = T_i + [t']$] ,
[$L_j = a$! $j=1,\cdots,N$] ,
[$U_j = u$, $U_j = u'$! $j=1,\cdots,N$] .

4.2.3 Inverse action

The inverse of an action can be easily defined.
e.g., The inverse action of $T_i = T_i + [t]$ is $T_i = T_i - [t]$.
The inverse action for an antecedent set is obtained by connecting all the inverse actions in the set. The following are the inverse actions corresponding to the antecedent sets shown in 4.2.2.

$T_i = T_i - [t]$,
$T_i = T_i - [t']$,
($L_1 \neg= a$) & $\cdots$ & ($L_N \neg= a$) ,
($U_1 \neg= u$) & ($U_1 \neg= u'$) & $\cdots$ & ($L_N \neg= u'$) .

4.3 Modification of grammar

Among the minimal antecedent sets for each rule, the optimal one is selected statistically using a corpus of text. Then, the grammatical rules are modified as follow. When the action part of a rule R' includes a member action of the antecedent set for a rule R, the action to activate R is added to the action part of R'. Likewise, when the action part of a rule R" includes the inverse action of the antecedent set for a rule R, the action to deactivate R is added to the action part of R".

We should add a comment on the status of a rule. In principle, a status is defined for each node. However, when the antecedent set is related to a node variable other than the key node variable, or an attribute relating to some inter-node actions, a status common to all the nodes is defined.

4.4 Improved grammar executor

An improved grammar executor which executes the modified grammar is illustrated in Fig. 6. A status table indicating the status of rules is introduced. It is updated by both the initializer and the transformer, and looked up by the controller. The initializer activates the rules in which the antecedent set includes an action in the process to create the initial object data. The transformer performs rule activating/deactivating actions included in the modified grammar. The controller looks up the status table when it selects the rule to apply. While the control is transferred to the pattern matcher if the rule is active, the controller immediately selects the next rule to apply if the rule is inactive.

5. Effectiveness

The improvement of processing efficiency by the proposed method is discussed from two points of view: the probability that rules are active and the overhead caused by dynamic rule activation.

(1) Probability that rules are active.
The probability that a rule succeeds in pattern matching is a lower limit for the probability that the rule is activated. However, the lower limit cannot be realized, because a rule is activated with prerequisite actions for its condition being satisfied. The state 'active' implies just the possibility that the rule will be applied successfully. The gap between the probabilities of 'active' and 'success' varies with rules. Fig. 7 illustrates two extreme cases. Fig. 7(a) is a case in which there is a minimal antecedent set for which occurrence probability is near the probability of the condition being satisfied. Fig. 7(b) is a case in which there is no such minimal antecedent set. As a matter of fact, (a) is a usual case and (b) is a rare case. A rule usually has a key condition featuring its relevant linguistic phenomenon, from which an effective antecedent set can be obtained. Therefore, the probability of 'active' is reduced to the same order as the probability of 'success'.

(2) Overhead of dynamic rule activation.
No additional conditions are introduced to the condition parts of rules to judge if an action to activate/deactivate a rule should be performed.

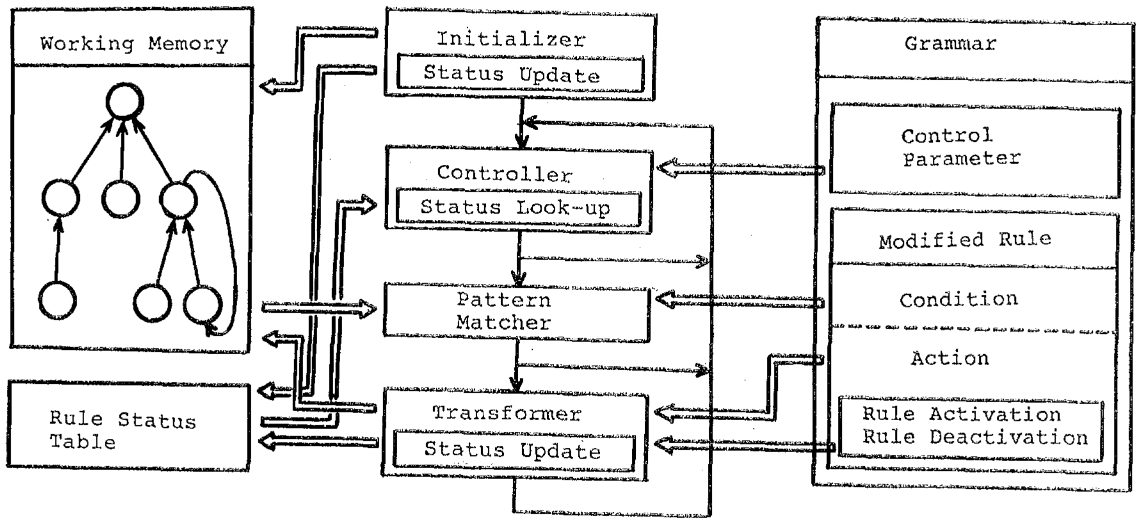
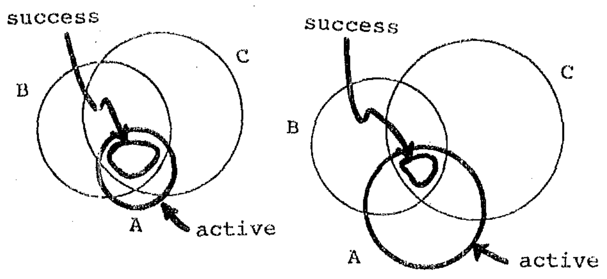


Fig. 6 Improved grammar executor



(a) Usual case (b) Rare case

A, B, C : minimal antecedent set

Fig. 7 Probability of 'active' vs. Probability of 'success'

Although rather a large number of actions to activate/deactivate a rule are added to action parts of rules, the action parts are infrequently performed. Moreover, although looking up the status of rules occurs frequently, its load is far smaller than that of pattern matching, which would be repeated if the dynamic rule activation were not used. Therefore, the overhead caused by dynamic rule activation can be neglected.

Another effect of the proposed method is that it can be applied to on-demand loading of rules when the memory capacity for a grammar is limited. That is, while rules with a large probability of 'active' are made resident on the main memory, the other rules are loaded when they are to be applied. Thus the frequency of loading rules is minimized.

6. Conclusion

An efficient execution method for rule based machine translation systems has been developed. The essence of the method is as follows. First, a grammar is pre-analyzed to determine an antecedent set for each rule. The antecedent set for a rule is a set of actions such that performing an action in it causes the possibility of the condition of the rule being satisfied, and the condition of the rule is unsatisfied if any action in it is not performed. At execution time, a rule is activated only when an action in the antecedent set for the rule is performed. The rule application is restricted to active rules. The probability of a rule being active is reduced to near the occurrence probability of its relevant linguistic phenomenon. Thus most pattern matching of unfruitful rules is avoided.

Acknowledgement: I would like to acknowledge Dr. Jun Kawasaki, Mr. Nobuyoshi Domen, Mr. Koichiro Ishihara and Dr. Tan Watanabe for their valuable advice and constant encouragement.

References

[1] Newell A. (1973). Production Systems: Models of Control Structures, in Visual Information Processing (ed. W. Chase; Academic Press).
[2] Boitet C., et al. (1982). Implementation and Conversational Environment of ARIANE 78.4, Proc. COLING82.
[3] Nakamura J., et al. (1984). Grammar Writing System (GRADE) of Mu-Machine Translation Project and its Characteristics, Proc. COLING84.
[4] Kaji H. (1987). HICATS/JE : A Japanese-to-English Machine Translation System Based on Semantics, Machine Translation Summit.
[5] Forgy C.L. (1982). Rete : A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem, Artificial Intelligence, Vol. 19.