ソフトウェア基礎論　16－9
プログラミング言語　4－9
（１９８６．２．７）

意味情報の推論的照合による言語変換方式の提案

渡辺　坦　（日立製作所　システム開発研究所）

内容梗概

言語変換においては、多重定義や多義性の解決、変換の質の向上等は、場合に応じた各種の技法を複合的に適用して行われてきた。本発表では、これらを統一的に扱える新しい方法を提示する。そこでは、入力文の列は属性構文木と呼ぶ意味属性つき抽象構文木に一旦変換される。そして、その部分木を変換規則で示された要素的属性構文木と意味情報も込めて比較し、一致するものがあれば変換規則で示される他の属性構文木で置き換える。基本的な変換規則と関係辞書とから、多数の変換規則が推論によって導出される。本方式は形式言語の変換ばかりでなく、自然言語の変換にも適用できる。

LANGUAGE TRANSFORMATION BY INFERENCIAL PATTERN MATCHING OF SEMANTIC INFORMATIONS

Tan Watanabe
Systems Development Laboratory, Hitachi, Ltd.

ABSTRACT

In language transformations, resolution of overloading and multivocality, and quality improvement of transformed results are usually achieved by applying many case-by-case techniques together. This article presents a new method to solve such probelms uniformly. Input sentences in source language is transformed to an attributed tree which is an abstract syntax tree with semantic relations attached to its nodes. A subtree of the attributed tree is compared with elementary trees specified in transformation rules requesting to satisfy semantic relations and it is replaced by other tree specified in matched rule. From a small set of basic rules and Relation Dictionary, numerous rules can be deduced by inference. This method is applicable both to formal language transformation and natural language transformation.

1. INTRODUCTION

Pattern matching is one of straightforward way to language transformations(1). Various compilers have been built by using syntactic patterns as templates for elementary transformation(2). In such compilers, semantic informations are mostly treated by action routines, because detailed semantic informations are difficult to be reflected to syntactic patterns(3). Attribute grammar(1) and case grammar(5) leads to a systematic treatment of semantic informations. Attribute grammar evaluator for such purpose search for syntactic pattern which matches with input sentence and transform it to a sentence in target language using semantic informations provided in the process of syntactic component contained in or containing the syntactic pattern. Semantic informations may be used also to resolve syntactic ambiguities(4).

This article presents a method of language transformation based not only on syntactic pattern matching but also on semantic pattern matching. Explosive increase in the number of patterns is supressed by introducing an inferencial capability in the process of pattern matching.

2. OUTLINE

A language transformer based on the method presented in this article consists of Input Analyzer, Transformer, Output Synthesizer, Transformation Dictionary, and Relation Dictionary as shown in Figure 1.

Input Analyzer transforms given input sequence of sentences to a tree structure representing the syntactic structure of the input sequence. Each node of the tree structure has some relations between attributes of components in the input sentences. Each relation takes the form of

$a \xrightarrow{r} b$ (b is related to a by a relator r).

A tree structure with such relations is called an attributed tree. The attributed tree generated from input sequence is called a primary tree in this article.

Relations visible from a subtree of an attributed tree are those ones which are attached to nodes to be encountered on the shortest way to the root node of the tree from the subtree. If a contradiction arise between such relations, then the relation attached to a node nearer to the subtree is the visible relation and the other one becomes a hinded relation.

In the following discussions, if a node x of a tree is on the shortest way to its root from a node y, then x is said higher than y and y is said lower than x. IF the node x is on the shortest way to its root from any nodes of a subtree s, then the node x is said to dominate the subtree s, and the subtree s is said to be subordinate to the node x.

Transformation Dictionary contains transformation rules showing the correspondence of elementary tree with secondary tree. Both of them are attributed tree. The elementary tree represents an elementary pattern of input language. Relations contained in the elementary attributed tree represent conditions to be satisfied during the matching process with input sequence. The secondary tree shows the result of transformation when the transformation rule is applicable.

Relation Dictionary contains relations in the form of

$a \xrightarrow{r} b$ (e.g. INTEGER $\xrightarrow{\text{range}}$(-214748348,214748347))

which represents a general relation satisfied in the world under consideration. A relation in the Relation Dictionary is visible from any attributed tree and its subtree if it does not contradict with visible relations in the attributed tree. If contradiction arose, then the relation attached to the root-ward node nearest to the subtree under consideration is the one to take effect. Relations in the Relation Dictionary may be considered to be attached to a pseudo-node located in higher position than the root node of the attributed tree.

Some relations may satisfy transitive law, that is,

if $a \xrightarrow{r} b$ and $b \xrightarrow{r} c$, then $a \xrightarrow{r} c$.

(e.g. X $\xrightarrow{\text{operationmode}}$COUNTER, COUNTER $\xrightarrow{\text{operationmode}}$INTEGER

then X $\xrightarrow{\text{operationmode}}$INTEGER.)

They are called transitive relators.

"Class" is a special transitive relator. For some relator r,

if $a \xrightarrow{\text{Class}} b$ and $b \xrightarrow{r} c$, then $a \xrightarrow{r} c$ holds.

(e.g., X $\xrightarrow{\text{Class}}$INTEGER and INTEGER $\xrightarrow{\text{Size}}$32, then X $\xrightarrow{\text{Size}}$32.)

Such relator is called an inheriting relator.

A relation induced by transitive relators and inheriting relators is called an infered relation. Sum of the number of successive transitions and the number of successive inheritance to induce a relation is called the number of inference steps to get the relation.

Transformer nestedly match the primary tree generated from given input sequence with elementary trees in the Transformation Dictionary. If some elementary tree matches with a primary subtree, a subtree of the primary tree, then the subtree is replaced by the secondary tree corresponding to the elementary tree. An

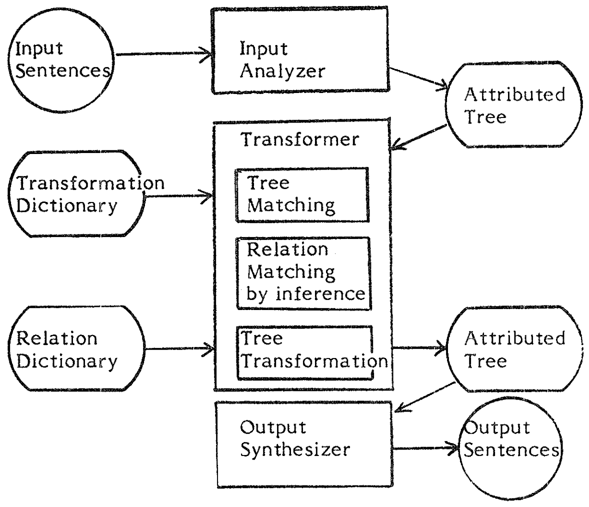


Figure 1. Inferential semantic pattern matching

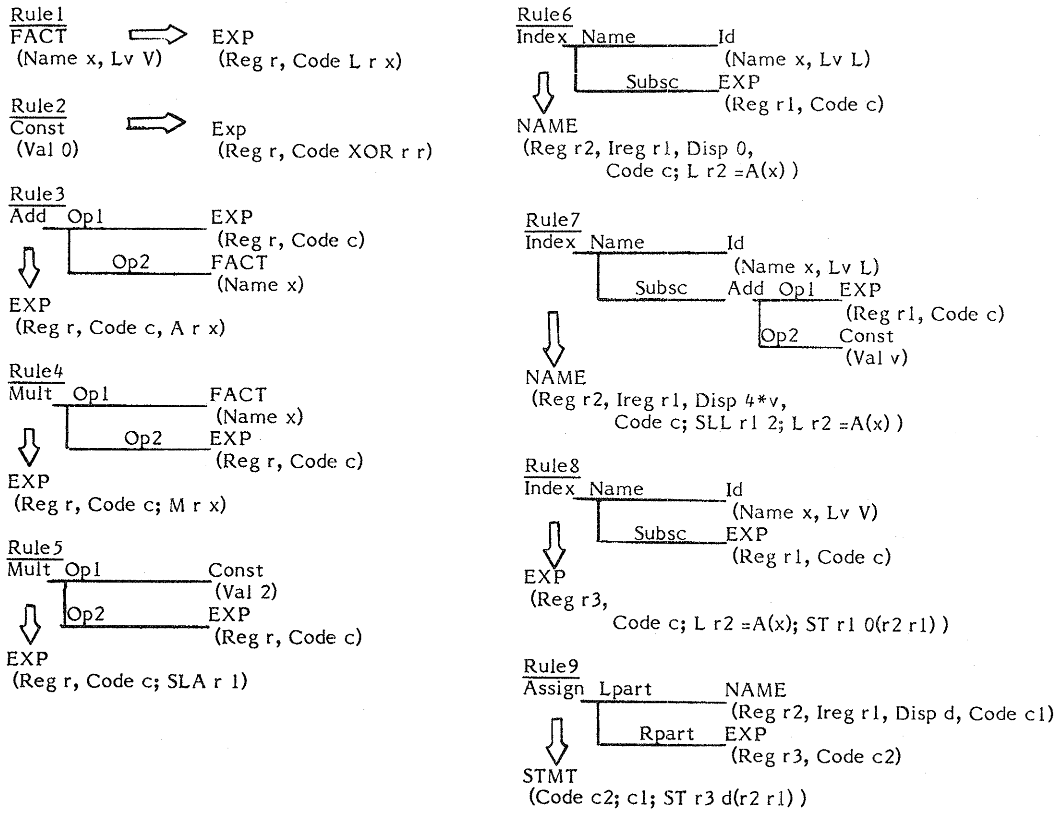

Rule1
FACT (Name x, Lv V)
EXP (Reg r, Code L r x)
Rule2
Const (Val 0)
Exp (Reg r, Code XOR r r)
Rule3
Add Op1 EXP (Reg r, Code c)
Op2 FACT (Name x)
EXP (Reg r, Code c, A r x)
Rule4
Mult Op1 FACT (Name x)
Op2 EXP (Reg r, Code c)
EXP (Reg r, Code c; M r x)
Rule5
Mult Op1 Const (Val 2)
Op2 EXP (Reg r, Code c)
EXP (Reg r, Code c; SLA r 1)
Rule6
Index Name Id (Name x, Lv L)
Subsc EXP (Reg r1, Code c)
NAME (Reg r2, Ireg r1, Disp 0, Code c; L r2 =A(x))
Rule7
Index Name Id (Name x, Lv L)
Subsc Add Op1 EXP (Reg r1, Code c)
Op2 Const (Val v)
NAME (Reg r2, Ireg r1, Disp 4*v, Code c; SLL r1 2; L r2 =A(x))
Rule8
Index Name Id (Name x, Lv V)
Subsc EXP (Reg r1, Code c)
EXP (Reg r3, Code c; L r2 =A(x); ST r1 0(r2 r1))
Rule9
Assign Lpart NAME (Reg r2, Ireg r1, Disp d, Code c1)
Rpart EXP (Reg r3, Code c2)
STMT (Code c2; c1; ST r3 d(r2 r1))

Figure 2. Transformation Dictionary for formal language

elementary tree is said to match with a primary subtree in the Transformation Dictionary if their tree structure match with each other and relations attacehed to the elementary tree are satisfied by relations attached to the subtree or by relations visible from the subtree, or relations infered from them.

Output Synthesizer generates a sequence of sentences in target language from the attributed tree transformed in the above way.

3. TRANSFORMATION PROCESS

More detailed explanation of the process of transformation follows. Input Analyzer transforms an input sequence to an abstract syntax tree, where each node represents a nonterminal or a terminal component of the input language, and each arc has indication of grammatical role of the subtree subordinate to it. Trivial differences such as slight difference in word-order and abbreviations uniquely amendable are resolved in the abstract syntax tree.

If node n has subtree s connected by an arc with grammatical indicator r, then it represents a relation

$n \xrightarrow{r} s.$

Syntactic relations can be organized into an abstract syntax tree. Some relations might not be represented by such abstract syntax tree. Input Analyzer analizes input sequence and extracts such semantic relations between its components and attach such relations of the form $a \xrightarrow{r} b$ to some node of the abstract syntax tree to make a primary tree. The selection of the node to attach a semantic relation is decided by a criteria that a relation should be attached to a lowest node which dominates all nodes where the same relation takes effect.

Transformer scans given primary tree to find some subtree which has the same tree structure with that of some elementary tree whose transformation rule is given in the Transformation Dictionary. The elementary tree may include a node which is gained by inference from the primary subtree. If such primary subtree is found, then relations attached to each node of the elementary tree is compared with the relations attached to corresponding node in the primary subtree. If every relations are the same, then the simplest success of pattern matching is attained. If some relation attached to a node in the elementary tree does not coincide with that of corresponding node in the primary subtree, then the relation is compared with relations visible from the node in the primary subtree. If same one is found in the visible relations, then the relation is said to be satisfied in the primary tree. If no same relation is found, then a trial is made to get the relation by inference uning the relations in the primary subtree and the relations visible from the primary subtree. If the trial succeeds then the relation is said to be satisfied in the primary tree by inference. If all relations in the elementary tree are satisfied in the primary tree, then the elementary tree is said to match with the primary subtree and the primary subtree is replaced by the secondary tree corresponding to the elementary tree.

A primary subtree which match with the elementary tree of some transformation rule may be different from the subtree containing all branches connected to the root node of the subtree. If some branch is not included in the primary subtree then the branch remains unchanged when thre primary subtree is replaced by the secondary subtree of some transformation rule. If more than one transformation are applicable, then the one with more branches in the elementary subtree has higher priority. Matching of branches nearer to the root of subtree has higher priority than that of branches farther to the root. If the number of matched branches is the same, then the transformation which requires fewer steps of inferences to decide its applicability has higher priority.

Transformation of tree is performed either in topdown method or in bottomup method. In both cases, resultant attributed tree obtained by the application of one transformation rule is used as the primary tree of other transformations so that transformations are repeatedly performred.

4.TRANSFORMATION OF FORMAL LANGUAGE

An example of transformation applied to object code generation from a source program is explained. In this example, the Transformation Dictionary contains rules shown in Figure 2, where secondary tree of each transformation rule is reduced to a leaf. A node or a leaf "a" with relation

$a \xrightarrow{r} b$

is written as

a (r b)

for brevity. Relation Dictionary contains the relations shown in (1) of Figure 3. Relator Lv specifies whether a storage location or a value is represented by the object to which Lv is attached. Lv is an inheriting relator.

Given input sequence is

P(2*I+1) := 0 ;

whose primary tree is shown in (3) of Figure 3. The primary subtree Id (Name P, Lv L) reduced to a leaf has the same tree structure as that of FACT (Name x, Lv L) of rule1 because the relations $\text{Id} \xrightarrow{\text{Class}} \text{NAME}$ and $\text{NAME} \xrightarrow{\text{Class}} \text{FACT}$ lead to a relation $\text{Id} \xrightarrow{\text{Class}} \text{FACT}$ by inference. But rule1 is not applicable to this primary subtree because the relation $\text{Id} \xrightarrow{\text{Lv}} \text{L}$ is not satisfied. The subtree consisting of the leaf node Const representing a constant 2 matches with rule1 because $\text{Const} \xrightarrow{\text{Class}} \text{FACT}$ and relation $\text{Const} \xrightarrow{\text{Lv}}$ V are given in the Relation Dictionary. In this case, however, the primary subtree beginning with Mult node matches with the elementary subtree of rule5 and this primary subtree is a larger subtree than the one consisting of the leaf node Const. Thus, rule5 has higher priority than rule1 and the subtree beginning with Mult is changed to

EXP (Reg r, Code L r1 I; SLL r1 1)

where, L r1 I is the object code generated by rule1 applied to the node Id (Name I). In order to apply rule1 to Id (Name I), the relation $\text{FACT} \xrightarrow{\text{Lv}} \text{V}$ should be satisfied. The relation $\text{EXP} \xrightarrow{\text{Lv}} \text{V}$ attached to the node Add is visible from the node Id (Name I). As Lv is an inheriting relator, relation $\text{Id} \xrightarrow{\text{Lv}} \text{V}$ is attained by sequence of relations $\text{Id} \xrightarrow{\text{Class}} \text{NAME}$, $\text{NAME} \xrightarrow{\text{Class}} \text{FACT}$, FACT $\underline{\text{Class}}$ EXP. The application of rule5 generates a subtree

$$\begin{tikzcd} \text{Add} \arrow[r, dash] \arrow[d, dash] & \text{EXP} \\ \text{Const} & \end{tikzcd}$$

, to which rule3 may be applicable. In this case, however, rule7 is applicable to the subtree beginning with Index which dominates the subtree beginning with Add. Thus, rule7 is applied and rule3 is not applied. As for Const (Val 0), both of rule1 and rule2 are applicable. The number of required inference steps of rule2 is smaller than that of rule1, hence, rule2 is applied to this Const subtree.

A sequence of Code attributes is generated during the process of such tree transformation. As a result, the input sequence is transformed to an assembly language program shown in (5) of Figure 3.

Consider the case where Transformation Dictionary does not contain rules 2, 5 and 7, which are special case rules corresponding to rules 1, 4, and 6 each respectively. In such case, rules to be applied to subtrees will be rule4 to Mult, rule3 to Add, rule 8 to Index, rule1 to Const, and resultant assembly language program will be the one shown in (4) of Figure 3. The resultant object code is 1 instruction longer and 6 bytes larger than the former result shown in (5) of Figure3.

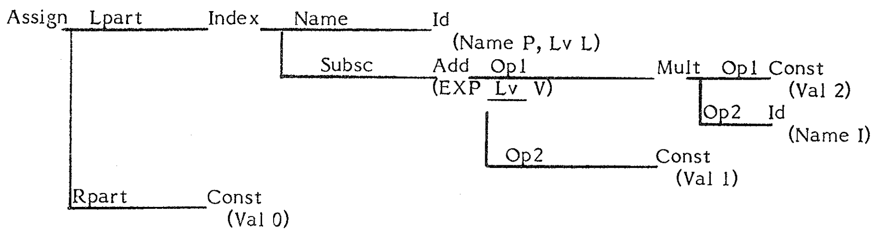
In this method, object code optimization will be specified by adding detailed transformation rules, each of which specifies a special case treatment of some general rule. No changes in general rules are required when detailed rules are added.

Id	$\xrightarrow{\text{Class}}$	NAME	Const	$\xrightarrow{\text{Class}}$	FACT
NAME	$\xrightarrow{\text{Class}}$	FACT	Index	$\xrightarrow{\text{Class}}$	NAME
FACT	$\xrightarrow{\text{Class}}$	EXP	Add	$\xrightarrow{\text{Class}}$	EXP
Mult	$\xrightarrow{\text{Class}}$	EXP	Assign	$\xrightarrow{\text{Class}}$	STMT
Const	$\xrightarrow{\text{Lv}}$	V			

(1) Contents of Relation Dictionary

P(2*I+1) := 0 ;

(2) Input sequence



(3) Primary attributed tree

```
L       r3,=0
L       r1,I
M       r1,=2
A       r1,1
SLL     r1,2
L       r2,=A(P)
ST      r3,0(r2,r1)
                (30 bytes)
```

(4) Result of translation
when rules 2, 5, 7 do not exist.

```
XR      r3,r3
L       r1,I
SLA     r1,1
SLL     r1,2
L       r2,=A(P)
ST      r3,4(r2,r1)
                (24 bytes)
```

(5) Result of translation
when rules 1 through 9 are available.

Figure 3. Example of formal language translation

5. TRANSFORMATION OF NATURAL LANGUAGE

As another example, a transformation from a sequence of Japanese sentences to a sequence of English sentences is explained. Let the sequence of input Japanese sentences be

私は計算機を動かそうとした。しかし、全然動かなかった。

watakushi wa keisanki wo ugokasouto sita。Shikasi zenzen ugokanakatta.

and let Transformation Dictionary contain rules shown in Figure 4, Relation Dictionary contains rules shown in (1) of Figure 5. A large attributed tree is generated by combining attributed trees, each of which is generated from a sentence in the input sequence. Informations to be passed from sentence to sentence are attached to a higher node dominating every subtree corresponding to a sentence which pass or refer the informations.

If rule 8 and 10 are not included in the Relation Dictionary, above Japanese sentences will be translated into

"I tried to move the computer. But it did not move at all."

assuming that omitted subject of the second sentence is the computer in the first sentence. Better translation can be produced when all relations shown in Figure 4 and (1) in Figure 5 are available, as it is described in the following. The translation of the word 私(watakushi) may result in either I or my. In this primary tree, rule2 is applied because the word is used as a subjective word and I is selected. The rules applicable to (ugokasu) in the first sentence are not only rule7 but also rule8. The applicability of rule8 is concluded by inference from the relation

計算機 (keisanki) Class 設備機械 (instrumental machine)

Rule8 has higher priority than rule7 because the elementary subtree of rule 8 has more relations satisfied by a primary subtree of input sequence than that of rule7, hence, rule8 is applied. As for 動く (ugoku) in the second sentence, rule10 is applied instead of rule9. This selection is performed in the same way as it is mentioned above.

The result of translation is

"I tried to run the computer. But it did not work at all.".

6. CONCLUSION

A method of language transformation based on inferencial pattern matching of abstract syntax tree with semantic informations is introduced. One example indicated its applicability to formal language compilation and the other example indicated its applicability to natural language transformation. Semantic informations relating several sentences are treated in the same manner as those contained in one sentence. The order of inference may affect the application order of transformation rules and a change in the order will result in the change of generated sentences in target language.

REFERENCES

1) Waite,W. and Goos,G.: Compiler Construction, Springer-Verlag, New York (1984).

2) Glanville,R.S. and Graham,S.: A new method for compiler code generation, Proc. of 5th Annual ACM Symposium on Principles of Programming Languages, pp231-240 (1978).

3) Farrow,R.: Experience with an attribute grammar-based compiler, Proc. of Conf. on Principles of Programming Languages, pp95-107 (Jan. 1982).

4) Watt,D.R.: Rule splitting and attribute-directed parsing, Semantics-Directed Compiler Generation, Lecture Notes in Computer Science 94, pp.363-392, Springer-Verlag, Berlin (1980).

5) Vinograd,T.: Languages as a Cognitive Process, Addison-Wesley, Reading, Massachusetts (1983).

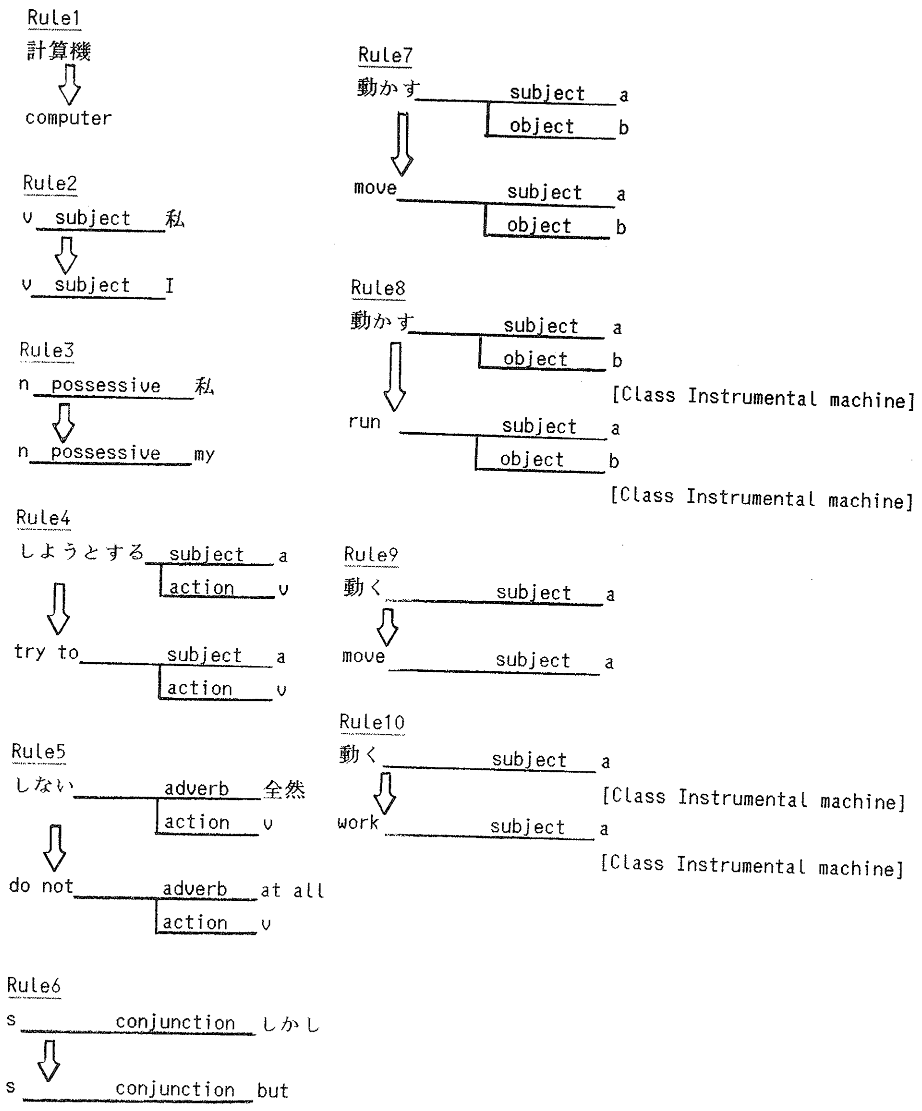
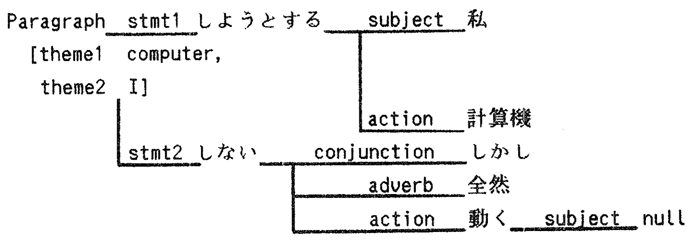


Figure 4. Transformation dictionary for natural language

私 $\xrightarrow{\text{Class}}$ Human
計算機 $\xrightarrow{\text{Class}}$ Instrumental machine
動く $\xrightarrow{\text{Class}}$ Action

(1) Relation Dictionary



(2) Primary attributed tree

I tried to move the computer. But it did not move at all.

(3) Translation result when rules 8 and 10 do not exist

I tried to run the computer. But it did not work at all.

(4) Translation result when all rules 1 through 10 are available

Figure 5. Transformation of natural language