COLING 82, J. Horecký (ed.)
North-Holland Publishing Company

A Parser Which Learns the Application Order of Rewriting Rules

Makoto Nagao, Jun-ichi Nakamura

Department of Electrical Engineering
Kyoto University
Sakyo, Kyoto
Japan

0. Introduction

The efficiency of syntactic analysis by using a set of rewriting rules is greatly influenced by the order or the arrangement of the rules for the application. There are some trials which subdivide the set of rules into subsets and specify the sequence of rule applications, thus avoiding the useless rule applications [1]. But the subdivision of the rule set and the specification of the sequence of rule applications are not so easy for the establishment of the most efficient analysis system.

We have developed a rewriting rule system which can manipulate arbitrary list of trees. The control mechanism of this system can adjust the weight of the rewriting rules, and can analyze the most plausible sentential structure f t, thus realizing the fast syntact analysis. The system learns (so to sp) the weight of importance of the rewriting rules during the analysis of input sentences.

1. Objectives of the Parser

We designed a new syntactic analysis system (a parser) with the following objectives.

(1) The function of rewriting rules must be powerful enough to handle a list of trees and to express transformational rules.
(2) All the possible sentential structures must be obtained for an input sentence in the sequence that the most plausible one is analyzed first.
(3) The analysis must be efficient enough for practical applications.
(4) The syntactic parser must have a learning mechanism as to the application sequences of rewriting rules to obtain the efficiency of analysis.

2. Method of Analysis

The input data for this parser is assumed as a word sequence which is the output of a morphological analysis. The output from this parser is a tree structure. The analysis is controlled by the best-first graph-searching technique about the rule applications.

2.1. Description of Rewriting Rules

The rewriting rules transform a list of trees into a list of trees. An example of the rewriting rule in this parser is shown in Fig. 1. It shows that if there is a symbol sequence composed of a tree not-V(erb), a tree N(oun) P(hrase), a tree C(ase-particle), and a tree not-A(dverbial)-P(article) in this order, this is transformed into a tree NP-C.

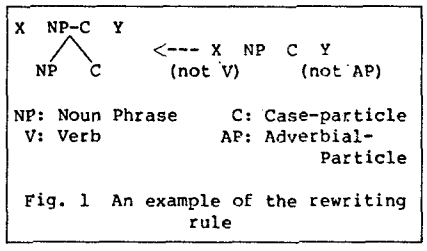


Fig. 1 An example of the rewriting rule

The right side of rewriting rules is a matching pattern which is to be found in the given input symbol string. Table 1 shows the function symbols to describe the matching patterns. By using these function symbols, it is possible to specify the repetition of pattern elements, to assign data to a variable, and so on. It is also

possible to check the input data by using user-defined functions. These functions enable us to describe complex syntactic functions, semantic relations, and many heuristic checks.

Table 1 Function symbols of the matching patterns

From	Function
?	Match an arbitrary tree.
#	Match any number of arbitrary trees.
(%F fn a1 ... an)	Evaluate function: fn whose arguments are a corresponding tree, a1, ..., and an. When the value is not NIL, matching succeed.
(%@ at x1 ... xn)	Match any number of lists of matching patterns x1 ... xn. Trees are assigned to variable at.
(%A x1 ... xn)	Matching succeeds if all x1, ..., xn are matched to a tree.
(%O x1 ... xn)	Matching succeeds if one of x1, ..., xn is matched to a tree.
(%N x)	Matching succeeds if x is not matched to a tree.

Table 2 Function symbols of the creation patterns

Form	Function
at	If at is a variable, then its value, otherwise at itself.
(*F fn x1 .. xn)	The value of the function: fn whose arguments are x1, ..., xn.
(*S at x)	The value of a generation of x assigned to the variable at.

The left side of rewriting rules is a creation pattern of new syntactic structures. Table 2 shows the function symbols for structure creation. User-defined functions can also be used to check certain relations in this creation pattern. We can generate an arbitrary tree structure by this rewriting rule system.

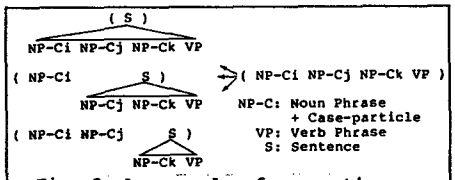


Fig. 2 An example of generating three left sides from a right side

As shown in Fig. 2, we can specify arbitrary numbers of structures in the left side for the same right side in a rewriting rule.

Each rewriting rule has a weight (basic score) and a function (fittedness function). The basic score is a static weighting measure which reflects the importance of a rule compared to the other rules of the same category. The basic score is adjusted by a learning process which will be explained in section 3. The fittedness function gives a dynamic weighting measure which expresses the fittedness of the rule application to a sentential structure. The function is a user-defined one which can use the data in both the right side and the left sides of the rewriting rules.

The basic score and the fittedness function are used for the sequence control of rule applications in the best-first graph-searching, which is the essential strategy to get the most plausible structural analysis first.

2.2. Flow of Analysis

Fig. 3 shows an intermediate structure in the course of a sentence analysis.

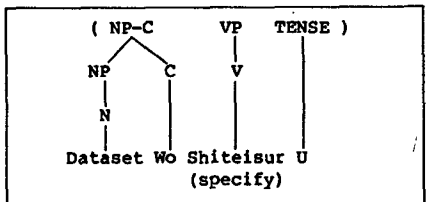


Fig. 3 The structure of a sentence during the analysis (LOT)

This structure is represented by a list of trees. We call this structure as a LOT (List of Trees). An analysis step is an application of a rewriting rule to a LOT as shown in Fig. 4. which changes the content of the LOT.

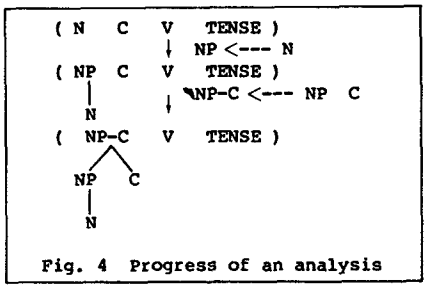


Fig. 4 Progress of an analysis

To obtain the result of an analysis one by one in the order of plausibility, we use the best-first graph-searching technique. If we regard a LOT as a node in a search graph, the new LOT created by the application of a rewriting rule to an old LOT is a sister node. When several rules are applicable to a LOT or the rule has several left sides, the same number of sister nodes are created from one mother node. The progress of analysis can be represented by an expansion tree (in general, by a graph) as shown in Fig. 5.

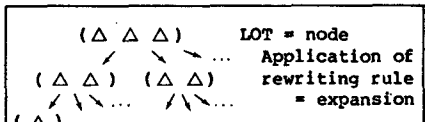


Fig. 5 Search tree

This tree can be regarded as a search tree. We expand the node which has the highest evaluation value (the score assigned to the LOT) first. The expansion is the application of a rewriting rule to a LOT. The evaluation value is obtained by the summation of the following four values:

(1) the evaluation value of the mother node.
(2) the basic score which is attached to the applied rule.
(3) the value obtained from the fittedness function which is attached to the applied rule.
(4) the score of the sentential pattern (SP: which will be explained in section 2.5), if it matches to the LOT.

Analysis is executed by principle of the best-first graph-searching technique as follows:

(1) Find the LOT which has the highest evaluation value.
(2) Apply rewriting rules to the selected LOT.
(3) If a rule is applicable, create new nodes (LOTs).
(4) Assign the new evaluation values to the new LOTs by the above method. (The initial LOT value is the summation of the scores attached to words.)
(5) Go to (1).

2.3. Application of Rewriting Rules.

The detail of the rule application sequence to a LOT which is selected by the best-first graph-searching technique is the following order:

(1) From left elements of the LOT.
(2) From the rule which has the longest right side.
(3) From the rule whose basic score is the largest.

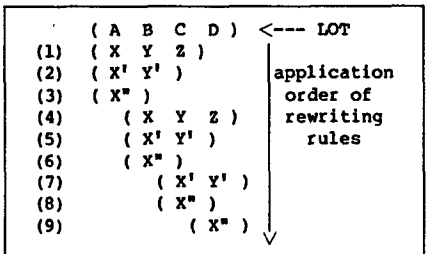


Fig. 6 An example of the application order of rewriting rules

Fig. 6 shows a simple example of the rule application when rewriting rules have (X Y Z), (X Y), and (X") as their right side, and (X"), and the selected LOT is (A B C D). First (A B C) is matched with (X Y Z). If the matching is not successful, (A B) is matched with (X' Y'). If the matching is not successful, (A) is matched with (X"). If the matching is not successful again, (B C D) is matched with (X Y Z), and so on.

To speed up the rule applications, matching patterns which are right sides of rewriting rules are reconstructed in a tree structure such as shown in Fig. 7.

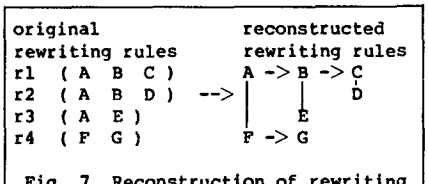


Fig. 7 Reconstruction of rewriting rules

In Fig. 7, if the first element of the LOT does not match with A, we do not need to test the rules r1 - r3. So the rule r4 alone is tested for the application. By this reconstruction, the number of rules which are to be applied to a LOT is decreased grately.

2.4. Pruning Rule

This parser is essentially a bottom-up parser, and there are cases that unnecessary expansions are executed. To minimize such unnecessary expansions, we introduced a mechanism of pruning such unnecessary nodes by certain pruning rules. For example, in the analysis of Japanese sentences, there must be some verb phrases (VP) to the right of a noun phrase (NP), so we use the pruning rule shown in Fig. 8. It matches with LOT, if LOT consists of some trees, a tree N, NP or NP-C, and trees which are not V, V-DA or VP in this order.

```
(# (%O (? N #) (? NP #) (? NP-C #))
   (%# NIL (%N (%O (? V #)
                   (? V-DA #)
                   (? VP #)))) )
```

(There must be V, V-DA or VP in the right of N, NP or NP-C.)

Fig. 8 An example of the pruning rule

The pruning rules are described by matching patterns just the same as the right side of rewriting rules. They are matched with the whole LOT at the time that a LOT is created. If a pruning rule matches with the LOT, the node is pruned.

2.5. Sentential Pattern

Sentential pattern (SP) expresses the global structure of a sentence. Fig. 9 shows examples of SP.

```
(1) (     S-OBJi      NP  V-DA  TENSE )  : -1
(2) ( NP-Ck   S-OBJj  NP  V-DA  TENSE )  : +1

          (a) Sentential patterns

(1) (      S-OBJi       NP  V-DA  TENSE )
          NP-Ck NP-Cm VPl TENSE
(2) ( NP-Ck     S-OBJj  NP  V-DA  TENSE )
                NP-Cm VPl TENSE

          (b) Corresponding LOTs

( NP-Ck  NP-Cm  VPl  TENSE  V-DA  TENSE )

          (c) Original LOT
```

Fig. 9 Examples of Sentential Patterns (SP)

The top two lines are the LOTs which are intermediate structures from an input sentence:

(NP-Ck NP-Cm VPl TENSE NP V-DA TENSE)
JSEUPDTE-program Ha Source-program-library Wo Shuseisuru(modify) Dataset-utility Dearu(is).
(JSEUPDTE program is a Dataset utility which modifies source program libraries.)

Each element of sentential pattern is a grammatical category name, not a tree structure. The elements of a sentential pattern are compared with the sequence of grammatical category names in a node. SP (1) represents that NP-Ck (JSEUPDTE-program-Ha) is related to VPl (the first embedded verb, Shuseisuru (modify)). SP (2) represents that NP-Ck is related to V-DA (main verb DA (is)).

The parser assigns SP-score and SP-rule to a sentential pattern. SP-score is a number such as shown in Fig. 9. This expresses the plausibility of the styles of sentences. In this example, SP (1) is assigned the numerical value: -1, and SP (2) is assigned the value: +1, as the SP-score. These two values mean that, when the main verb is V-DA, the first NP-C has a tendency to be related to the main verb rather than to the first embedded verb. This SP-score is added to the evaluation value explained in section 2.2. Therefore, analysis (1) takes precedence over analysis (2) in this case.

```
( NP-C  NP  C  VP  TENSE )  : SP
   ( ( rule-1 2 )
     ( rule-2 1 )     : SP-rule
     ( rule-3 1 )
     ( rule-4 1 ) )
```

Fig. 10 An example of SP-rule

Fig. 10 shows an example of SP-rule. The sentential pattern whose SP-score is positive has at least one correct analysis. And a sequence of rule applications to the sentential structure is guaranteed. SP-rules represent this sequence. However, it is not evident whether the sentential pattern whose SP-score is negative has correct analyses, because it has at least one incorrect analysis. So we do not attach any SP-rule to it.

SP-rule in Fig. 10 shows that we can get a correct analysis, if we apply rule-1 - rule-4 to the LOT. Fig. 11 shows this process of rule applications. The sequential rule application of these four rules is equivalent to a

rewriting rule shown in Fig. 12. But the rewriting rules of the form shown in Fig. 10 are much better because the semantic check functions can be easily introduced to the simpler rules such as those in Fig. 10 rather than to such complex rules as those in Fig. 12.

```
( NP-C  NP  C  VP  TENSE )
          | ---- NP-C <--- NP  C ( rule-1 )
( NP-C  NP-C  VP  TENSE )
        | ---- S-NUCL <--- NP-C  NP-C  VP ( rule-2 )
( S-NUCL  TENSE )
      | ---- S-OBJ <--- S-NUCL TENSE ( rule-3 )
( S-OBJ )
   | ---- S <--- S-OBJ ( rule-4 )
(  S  )
```

Fig. 11 An example of the SP-rule application

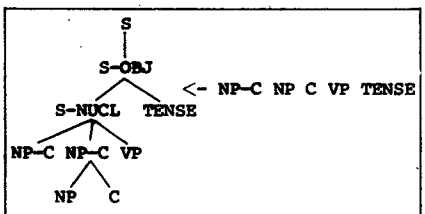


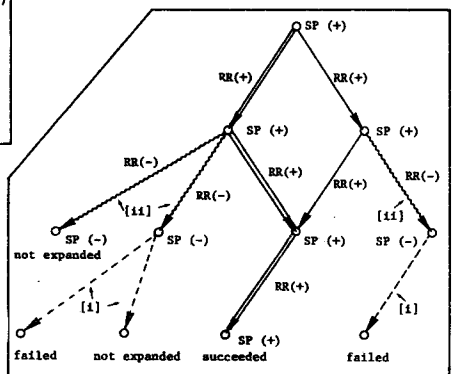
Fig. 12 An example of the equivalent rewriting rule

Each LOT is compared to sentential patterns from the first element of the LOT. The LOT is regarded as matched if the first part of the LOT matches a sentential pattern.

3. Supervised Learning of Basic Scores, SP-scores and SP-rules

To increase the efficiency of the analysis, the parser controls basic scores attached to rewriting rules, SP-scores and SP-rules. It is not easy for rule writers to assign scores to rewriting rules and to sentential patterns, and also to write SP-rules for a sentential pattern. We tried to adjust these scores and to get SP-rules by the supervised learning in which the user teaches the correctness of an analysis to the parser.

Fig. 13 shows an example of a search graph when a sentence is analyzed. Each node of the search graph corresponds to a LOT. Each arc corresponds to a rule application. We can regard the LOTs on the path from the root node to the successful node as useful structures, and the rewriting rules on the path as useful rules for the future analysis of similar sentential structure. On the other hand, other LOTs and rewriting rules in the search graph are regarded as useless to the future usage. But the nodes and arcs [i] in Fig. 13 are not the direct reason of the failure. The direct cause for the failure comes from the nodes and arcs [ii] in Fig. 13.



RR: Rewriting Rule
SP: Sentential Pattern

Fig. 13 Relation between the state of the expansion and failure or success of the analysis

The parser changes the scores of rewriting rules and SP-scores in the following way:

(1) Increase the scores of the rewriting rules and SP-scores on the path from the root node to the successful node, and those on the pathes which flow into the successful pathes.
(2) Decrease the scores of the rewriting rules and SP-scores on the first arcs of the pathes which flow out the successful pathes.

SP-rules are gathered for each sentential pattern on the successful pathes by using the information in the search graph.

4. Result of Some Experiments

The sample sentences to be analyzed are taken from a computer manual in Japanese. About 150 sentences are used for the experiments. Conjunction structures of noun phrases are eliminated from these sentences. Among

150 sentences, 20 sentences are used for the supervised learning. These are selected randomly. The rewriting rules are created from the grammar proposed by Okutsu [2]. The number of rewriting rules is 54. The rewriting rules in this experiment do not have the semantic check functions for simplicity. They are prepared to get the syntactic structures for a sentence.

4.1. Experiment 1 - Learning of Basic Scores of Rewriting Rules.

To see the efficiency improvement of the analysis from the contribution of basic scores, SP-scores and SP-rules are not used. The initial order of the rewriting rules is determined by random numbers. The initial basic scores are set the same value 1 for all rules. We adjusted basic scores 4 times, every time after 20 sentences for learning are analyzed. We compared the CPU-times of the 2nd, 3rd and 4th analyses to the CPU-time of the 1st analysis. The result is shown in Table 3.

Table 3 Effect of basic scores

	2nd/1st	3rd/1st	4th/1st
max.	99.37%	102.10%	108.78%
average	94.62%	96.75%	96.47%
min.	87.69%	87.88%	89.49%

(The values are the ratio of the CPU-time per word.)

Table 3 tells us that the basic scores of rewriting rules are not so useful for the improvement of the efficiency of analysis. The learned order of rewriting rules does not have a significant tendency. The reason is that the structure of natural languages is recursive and the relative order of rules are more important to the analysis than the over-all ordering, so that the basic scores cannot express the relative order.

4.2. Experiment 2 - The Effect of SP-scores and SP-rules

The learning of the SP-scores and SP-rules are done by analyzing the set of sample sentences once (20 sentences selected among 153 sentences randomly). Then the analysis of the set of sample sentences (153 sentences) is done with and without using SP-scores and SP-rules. The result of the experiment is in Table 4.

Table 4 Effect of SP-scores and SP-rules

	the same SP	not the same SP
number of sentences	42	111
max.	26.06%	108.63%
average	19.23%	67.36%
min.	1.03%	9.46%

(The values are the ratio of the analysis time with SP-scores and SP-rules to the analysis time without them.)

About 200 sentential patterns are extracted from the 20 sample sentences for learning. SP-rules are very useful for the sentences which have the same sentential patterns, because the rewriting rules and their application sequence in the analysis of the sentential pattern can be obtained from SP-rules which are defined from the past analysis, and no more trial search is necessary. 27.5% of sample sentences have the same sentential patterns as the sentences for learning. This means that some documents like a computer manual contain very similar sentences. Sentential patterns and SP-rules are useful for the analysis of such documents.

5. Conclusion

The experiments to examine the effect of learning are performed. The results of the experiment shows that SP-rules are very useful. This means that this parser can learn the style of the sentences and can increase the efficiency of analysis when the sentential structures of the texts in the particular field are restricted.

This parser is implemented by LISP on FACOM M-200 in Computer Center of Kyoto University.

References

[1] Boitet, C., Automatic Production of CF and CS-analysers using A General Tree-Transducer, Université scientifique et médicale de Grenoble, 1979.

[2] Okutsu, K., Seisei Nippongo Bunpo-ron, Taishukan-shoten, 1974.