A Bottom-up Parser where Entire Operation is Conducted in the Letter String Region

Hiroshi Sakaki
Faculty of Informatics
Meisei University
2-590 Nagafuch Ohme-shi Tokyo
Tokyo 198-8655 JAPAN

Takashi Tanaka
SCC Corporation
5-36-14 Nakano Nakano-ku
Tokyo 164-0001 JAPAN

Michihiko Seki
Pro-log Corporation
3-14-2-207 Amanuma Suginami-ku
Tokyo 167-0032 JAPAN

Abstract: - This paper treats a natural language parser of bottom-up type. The characteristics of the parser lies in that the data treated keep the shape of letter string through the entire parsing operations. Letter strings including parentheses express the partial trees generated in the course of parsing. This expression helps to avoid list expression usually used to represent trees.

Key-Words: - parser, bottom-up, letter string, machine translation, LINGOL, C language, OR node

1 Introduction

This paper concerns about chart parser proposed by Martin Kay[1]. A chart parser utilizes a chart composed of nodes representing word borders of input sentence and arcs connecting those nodes. To each arcs, the analysis result or intermediate structures produced during analysis are tied.

LINGOL is a control method of bottom-up parser. This parser is improved into extended LINGOL[2][3]. On these basis, a parse control method in which plural analysis results are bundled by special kind of node called OR node is proposed. The bundling is effective for saving memory and calculation task. A large scale MT system called KATE that uses this bundle method has been constructed. Here in this paper, parsing method of this type is called KATE type parsing method[4].

Recently various types of parsers are proposed. The ones similar to this paper's method are Morawietz's, and Watanabe's that utilize context information but do not treat data in letter string region[5][6]. There are many types utilizing stochastic information [7].

The parser proposed in this paper conducts processing of intermediate structures entirely on letter string region and still conforms to the KATE type control method. The processing on letter string region simplifies greatly calculation mechanism of parser.

2 Operation applied to partial trees

This chapter illustrates the operation applied to partial trees tied to arcs.

The method of this paper uses 3 procedures called procedure 1', procedure 1 and procedure 2, as fundamental procedures. The following is the description of these procedures.

KATE type parsing method deals with 2 kinds of arcs one of which is active arc and the other of which is inactive arc. An active arc represents a tree having unsaturated nodes and an inactive arc represents a complete tree. At an inactive arc, the kinds of "category" to fill the unsaturated terms are designated. The word "category" expresses grammatical category including, in this paper, word names. The section of existence of an arc on a chart is called "span". The expression that the arc "spans" the section is also used.

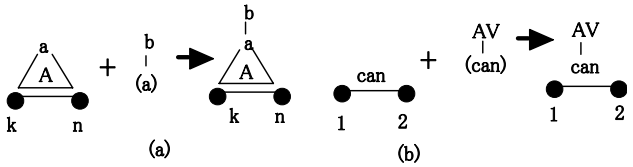

Fig.1 Operation of procedure 1'

A context-free rewriting rule has one left-hand side term and one or more right-hand side terms. The graphical expression of a rewriting rule is a tree composed of one upper node corresponding to the left-hand side term and one or more lower nodes corresponding to the right-hand side terms. Because of above correspondence, the left-hand side term of rewriting rule is called upper node and right hand side terms are called lower nodes.

The tree tied with an arc is said to be the label of the arc. In this paper the terms "arc" and "label of arc" designate the same object. Procedure 1' is the procedure where a rewriting rule with single lower node is applied to a inactive arc giving rise to a new inactive arc. Fig.1 shows the application of procedure 1'. Fig.1(a) is generic expression of the process and Fig.1(b) illustrate an example of the process. The

portion expressed by a triangle denotes a tree structure. This tree structure has the name "A" and the topmost node of the tree structure is the node "a".

First, Fig.1(a) is investigated. The portion to the left of "+" sign is the inactive arc and the portion to the right of "+" sign is the rewriting rule. The lower node of rewriting rule is "(a)" indicating that the category to fill the term is "a". The portion to the left of the arrow is situation before the application of rewriting rule and portion to the right is the result of application.

At procedures investigated hereafter, only the topmost node of a tree or an inactive arc can fill the term of a rewriting rule or an active arc. As a consequence, the category of the topmost node of an inactive arc is called the category of the inactive arc.

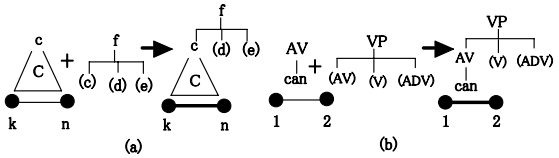

Fig.2 Operation of procedure 1

Procedure 1 is the procedure where a rewriting rule with more than one lower nodes are applied to a inactive arc giving rise to a new active arc. Fig.2 shows the application of procedure 1. Fig.2(a) is generic expression of the process and Fig.2(b) illustrate an example of the process.

The expression to the left of the "+" sign shows that the tree with topmost node "c" is the label of the arc spanning the section between nodes k and n. The portion to the right of "+" sign is the rewriting rule with 3 lower nodes.

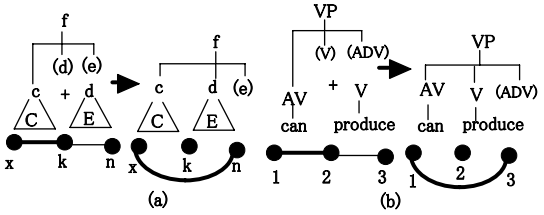

Fig.3 Operation of procedure 2

Filling in of unsaturated lower nodes proceeds at the order of left to right. The lower node "(c)" shows that this node must be filled with the category "c". The application result is the tree construction representing an active arc situated to the right of the arrow. In this case the nodes "(d)" and "(e)" are unsaturated nodes characteristic of an active arc. As the fixed portion at the lower portions in the application result spans the section between node k an n, the span of the active arc obtained is between nodes k and n. In this paper, thin lines express inactive arcs and thick lines express active arcs. Fig.2(b) shows an example of application of procedure 1.

Procedure 2 is the procedure where the unsaturated term of an active arc existing to the left of boarder node is filled with the topmost node of an inactive arc existing to the right of the boarder node, generating an arc. If the label of a newly produced arc has unsaturated term, it become an active arc, otherwise it become an inactive arc.

Fig.3(a) is generic expression of the process. The portion to the left of "+" sign is the active arc and portion to the right is inactive arc. The border node is node "k". The lower node "(d)" of the active arc shows that the category to fill the term is category "d".

The application of procedure 2 generates an active arc still having unsaturated term "e". This newly generated tree has fixed portions spanning a section between node x and node n. So the newly created active arc spans between node x and n.Fig.3(b) is an example of the application of procedure 2.

Aggregation is the operation where plural inactive arcs having identical span and identical topmost node is bundled using special kind of node called node "OR". The structure obtained by the application of aggregation has the same span as the input of the operation.

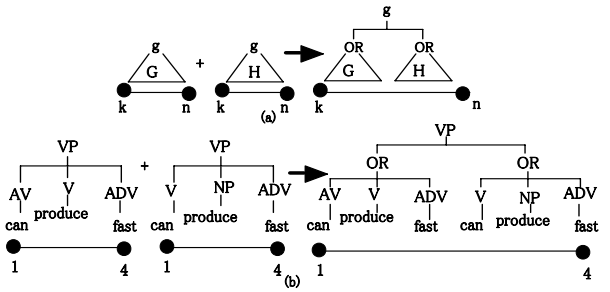

Fig.4 Operation of aggregation

Fig.4 is generic expression of aggregation operation. The label of 2 inactive arcs are trees G and H. As these 2 trees have identical span and topmost node, an inactive arc is generated by aggregation operation. The label of newly generated inactive arc is shown to the right of the arrow. The original topmost nodes are replaced by the OR nodes and, successibly, OR nodes is bundled by the original topmost node. The span of aggregation result is the same as that of original arcs. After the aggregation operation, the original arcs are deleted. The aggregation saves memory and calculation task in parsing. Fig.4(b) is an example of aggregation operation.

3 Flow chart to control parsing

Fig.5 contains the example sentence used to illustrate the operation and rewriting rules to analyse the sentence. Fig.5(1) through 5(9) are respectively rewriting rules and Fig.5(10) is input sentence. The lower nodes in rewriting rules are enclosed by

parenthesis to show that these nodes request the category in the parenthesis.

Fig.6 is the flow chart of the parser controling operation mentioned in last chapter. The parser proposed in this paper comforms to KATE type parsing method and consequently, the control flow chart is shaped after KATE system. The characteristic of parser of this paper, never-the-less lies in that entire tree structural operation is conducted in letter string ragion.

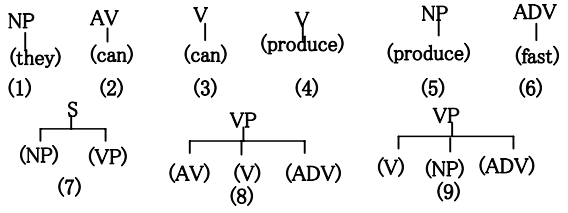

Fig.5 Input sentence and rewriting rules

Descriptions for the portions composing the flow chart is made here. Each portion of the flow chart has its name and acronym. The portions which manipulate the partial trees are "Process1'(P1)" for the operation of procedure 1', "Aggregation(AG)" for aggregation, "Process1(P2)" for prcedure 1, "Process2(P3)" for procedure 2. Partial trees are pieces of analysed tree for input sentence. The portions which controls existence of arcs are "Inactive arc erasure(EA)" erasing entire inactive arcs from the chart and "Word input(WI)" introducing an inactive arc composed of single word to the chart. The right end of the inactive arc is set to the value of "n" parameter.

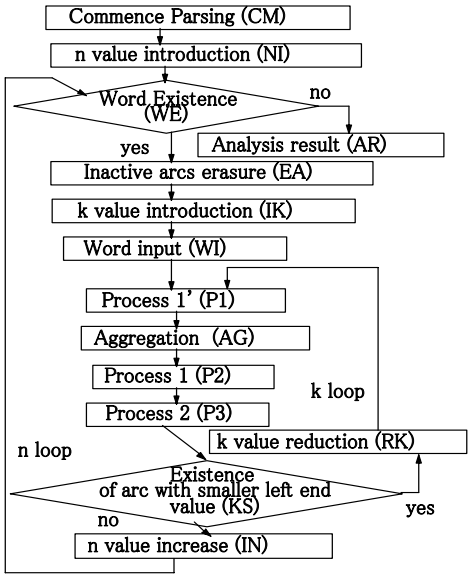

Fig.6 Flow chart of parser

Parameters "k" and "n" controls the operation of the flow chart of Fig.6. Inactive arcs processed at the portions P1, AG, P2 and P3 are limited to those having span between n(left end) and k(right end). There is no limitation for span of active arc.

Portions directing action of the parser are these 3 portions. "Commence parsing(CM)" initiate the action of parsing. "Word existence(WE)" detects the existence of word not introduced to the chart. If the decision is "absent of word" the operation turns to the "Analysis result(AR)". If decision is "present" the operation continues on to EA. AR generates the analyzed tree.

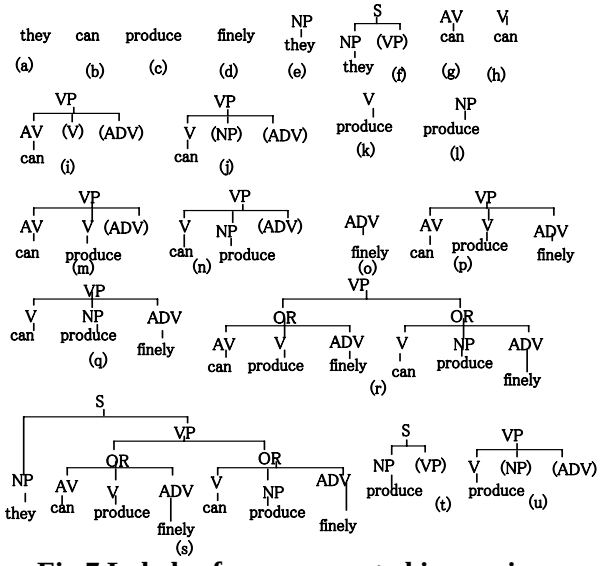

Fig.7 Labels of arcs generated in parsing

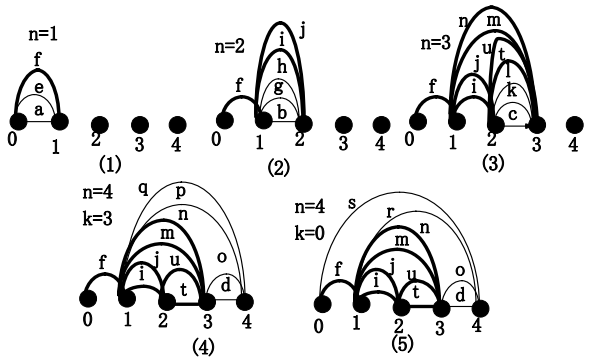

Fig.8 Arcs generated in parsing operation

Following portions concerns about designation of parameter "k" and parameter "n". Portion of "n value introduction(NI)" sets n value to 1 at the commencement of parsing. Portion of "k value introduction(IK)" sets the k value to n-1. Portion of "Existence of arc with smaller left end value(KS)" detects the existence of inactive arc having left end whose position number is smaller than the present value of "k". If there is no such inactive arc, the operation shifts to portion of "n value increase(IN)"

that increases the "n" value by 1. If such inactive arcs exist, operation turn to portion of "k value reduction(RK)" which reduces k value to the left end value of the inactive arc whose left end is leftmost among those of inactive arcs.

The following is the survey of the parsing process comforming to the flow chart in Fig.6 which utilizes 3 procedures shown in the last chapter. The case where the sentence in Fig.5(10) is input and rewriting rules in Fig.5(1) through (9) are used, is examined.

The inactive arcs and active arcs generated during parsing process are displayed at the figures in Fig.8. The labels of these arcs are illustrated in Fig.7. Here, the label belonging to an arc is a tree in Fig.7 having the sub-number identical to the arc name.

The operation is expressed by the series of portions in the flow chart of Fig.6 succeeded by the explanation for the series. The transfer among portions in the series is expressed by the sign "-->". Description of each element of the series is called "action" and is composed of the acronym of the portion follwed by values of "k" and "n" parameters enclosed by parenthesis. When the value of a parameter is not determined, the sign "-" is used instead. It must be noticed that parameters "k" and "n" decides the ends of inactive arcs treated by this parser.

The tracking of the operation is divided into blocks the borders of which are the erasure of inactive arcs by the portion EA. Each figure in Fig.8 corresponds to respective block.

The first block operation is: CM(-,-) --> NI(-,-) --> WE(-, 1) --> EA(-,1) --> IK(0, 1) --> WI(0,1) = generation of a --> P1(0,1) = generation of e --> AG(0,1) --> P2(0,1) = generation of f --> P3(0,1) --> KS(0,1) --> IN(0,1) --> WE(0,2) --> EA(0,2) = erasure of a and e.

The description after the symbol "=" shows operation on arcs. The chart for this block is Fig.8(1). This chart is composed of 4 nodes which are nodes 0, 1, 2, 3 and 4. The 3 nodes at right side are isolated because no arcs connecting them exists. At the action WI(0,1), the inactive arc "a" shown in Fig.7(a) is introduced to the chart.

At action P1(0,1), procedure 1' is applied to the inactive arc "a" and the rewriting rule in Fig.5(1) giving rise to the inactive arc "e" having the span identical with arc "a". At action P2(0,1), procedure 1 is applied to the inactive arc "e" togetner with rewriting rule in Fig.5(7) giving rise to the active arc "f". By the action EA(0,2), entire inactive arcs in Fig.8(1) are eliminated.

The second block operation is: IK(0,2) --> WI(1,2) = generation of b --> P1(1,2) = generation of g and h --> AG(1.2) --> P2(1,2) = generation of i and j --> P3(1,2) --> KS(1,2) --> IN(1,2) --> WE(1,3) --> EA(1,3) = erasure of b, g and h.

The chart for this block is Fig.8(2). At the action WI(1,2), the inactive arc "b" is introduced to the chart. At action P1(1,2), procedure 1' is applied to the inactive arc "b" together with the rewriting rule in Fig.5(8) giving rise to the inactive arc "g". Similarly, inactive arc "h" is obtained from "b" using the rewriting rule in Fig.5(9). At action P2(1,2), procedure 1 is applied respectively to the pair of inactive arc "g" together with rewriting rule in Fig.5(8) and to the pair of inactive arc "h" together with rewriting rule in Fig.5.(9), generating respectively the active arcs "i" and "j". By the action EA(1,3), entire inactive arcs in Fig.8(2) are eliminated.

The third block operation is: IK(1,3) --> WI(2,3) = generation of c --> P1(2,3) = generation of k and l --> AG(2,3) --> P2(2,3) = generartion of t --> P3(2,3) = generation of m and n --> KS(2,3) --> IN(2,3) --> WE(2,4) --> EA(2,4) = erasure of c and k..

The chart for this block is Fig.8(3). At the action WI(2,3), the inactive arc "c" is introduced to the chart.

At action P1(2,3), procedure 1' is applied to the inactive arc "c" together with rewriting rule in Fig.5(4) generating the inactive arc "k". In addition, inactive arc "l" is obtained from "c" using the rewriting rule in Fig.5(5). At P2(2,3), application of procedrue 1 to the inactive arc "l" with the use of rewriting rules of Fig.5(7) and Fig.5(9) produces active arcs "t" and "u". These active arcs do not contribute to the generation of analysis result. At action P3(2,3), procedure 2 is applied respectively to the pair of inactive arc "k" and active arc "i" and to the pair of inactive arc "l" and active arc "j", generating the active arcs "m" and "n".

By the action EA(2,4), entire inactive arcs in Fig.8(3) are eliminated.

The fourth block operation is: IK(2,4) --> WI(3,4) = generation of d --> P 1 (3,4) = generation of o --> AG(3,4) --> P2(3,4) --> P3(3,4) = generation of p and q --> KS(3,4) --> RK(3,4) --> P1(1,4) --> AG(1,4) = generation of r and elimination of p and q.

The chart for this block is Fig.8(4). At the action WI(3,4), the inactive arc "d" is introduced to the chart. At action P1(3,4), procedure 1' is applied to the inactive arc "d" together with rewriting rule in Fig.5(6) generating the inactive arc "o".

At action P3(3,4), procedure 2 is applied to the inactive arc "o" and active arc "m" generating the inactive arc "p". Similarly, inactive arc "q" is obtained from "n". KS(3,4) decides that next action is RK(3,4) because there are inactive arcs "p" and "q" having the left end at position 1 which is amaller than present value of "k" which is 3. The action RK(3,4) reduces the value of "k" to 1 which is the left end position of

the arcs "p" and "q". The inactive arcs between positions 1 and 4 are hereafter treated. At action AG(2,4), inactive arcs "p" and "q" are aggregated into inactive arc "r". Then inactive arcs "p" and "q" are deleted.

The fifth block operation is: P2(1,4) --> P3(1,4) = generation of "s" --> KS(1,4) --> RK(1,4) --> P1(0,4) -->P2(0,4) --> P3(0,4) --> KS(0,4) -->IK(0,4) -->WE(0,5) -->AR(0,5) = completion of parse with the output "s".

The chart for this block is Fig.8(5). At action P3(0,4), procedure 2 is applied to the inactive arc "r" and active arc "f" generating the inactive arc "s". This is analysis result aimed at that is generated from the input sentence in Fig.5(10).

Further operation make completion of parsing. Finally, action WE(0,5) decides that entire word in input sentence has been introduced to the chart and turns, it decignates that next action is action AR(0,5). AR(0,5) yield the inactive arc "s" as output.

4 Treatment of tree structure in letter string region

4.1 Expression of tree structure by letter string

Tree structure is expressed by letter string in the way of Fig.9. Fig.9(a) shows generic illustration of the method. The upper part is the tree structure and the lower part is its letter string expression. As seen in Fig.9(a), the letter string consists of a parenthesis pair in which the topmost node and child tree structures are placed. The topmost node is placed directly after the open parenthesis and child tree structures are arranged keeping the order of existence at the tree structural expression.

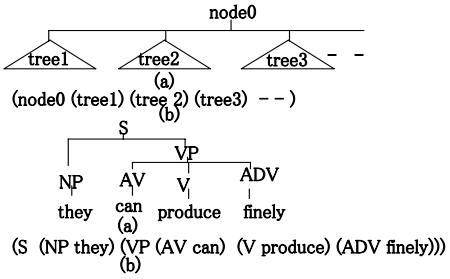

Fig.9 expression of tree structure by letter string

The node0 being the topmost node has no structure taking the characteristics of "atom", whereas child tree structures have constructions. Namely, each child structures have its own letter string at which the first and last letters are parenthesis. Never-the-less, the regulation is that the "atom" structure at tree structural expression is expressed without parenthesis.

Fig.9(b) is an example of letter string expression where the upper part is tree structure and lower part is its letter string expression. In this paper, only lower case letters expresses word nodes.

4.2 Expression of procedures in letter string region

Here, the method to deal with parsing procedures in the region of letter string, is investigated. Fig.10 is the method to treat procedure 1' in the ragion of letter string. Fig.10(a) is the tree structural expression of procedure 1' which is identical with Fig.1(b) except for little modification. Here, procedure 1' is applied to the inactive arc with the label composed of a node "can" together with a rewriting rule requiring the node "can", generating the structure to the right of the arrow. The input and output of the procedrue have the same span.

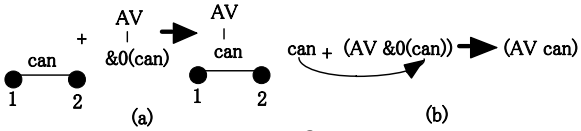

Fig.10 Treatment of procedure 1' in the ragion of letter string

Position number is composed of a letter "&" followed by a digit. Position numbers are attached to the lower nodes of rewriting rules. In Fig.10(a), a position number "&0" exists. The position number of a node is identical with the relative order (counted from 0) of the node among sister nodes. Position number is necessary for the treatment in letter string region. By the position number, the designation of unsaturated condition and its order among sister nodes are expressed.

Fig.10(b) shows the process to conduct the procedure 1' operation of Fig.10(a) in the region of letter string. As the correspondence of the portions appearing in Fig.10(a) and Fig.10(b) is obvious, no description is made for the correspondence.

The portion to the left of "+" sign is a letter string expressing of a inactive arc. The way of expression comforms to the method in Section 4.1. Hereafter, all the letter string expressions are defined to comform to the method.

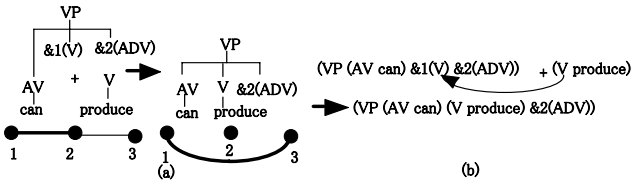

Fig.11 Treatment of procedure 1 in the ragion of letter string

The portion to the right of "+" sign in the Fig.10(b) is letter string expression of the rewriting rule the lower node of which requires node "can". The procedure 1' is conducted by "over-writing" the unsaturated node compoosed of position number(&0) and the required

node(can) with the inactive arc. The operation generates the letter string to the right of the arrow which is letter string expression of the structure in Fig.10(a).

It must be noted that the information of span of an arc exists in separate portion. The information of span is attached to the inactive arc and carried over to the result of the operation of procedure 1'.

Fig.11 is the method to treat procedure 1 in the region of letter string. The example shown in Fig.11 is the same as that in Fig.2. Fig.11(a) is tree structural expression and Fig.11(b) shows letter string expression.

The operation of procedure 1 is the same as that of procedure 1' except for the fact that the number of lower nodes of rewriting rule is more than 1.

The procedure 1 is conducted by "over-writing" the unsaturated node compoosed of position number(&0) and the required node(AV) with the inactive arc.

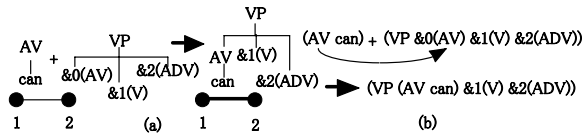

Fig.12 Treatment of procedure 2 in the region of letter string

Fig.12 is the method to treat procedure 2 in the region of letter string. The example shown in Fig.12 is the same as that in Fig.3. Fig.12(a) is tree structural expression and Fig.12(b) shows letter string expression. This procedure connects an active arc situated to the left and an inactive arc situated to the right.

The procedure application is conducted by "over-writing" the unsaturated node compoosed of position number(&1) and the required node(V), with the inactive arc having topmost node "V". To accomplish the procedure application, the condition that the right end of the active arc is identical with the left end of the inactive arc must be met. This condition is investigated by the span information attached to the arcs.

Each active arc has the position number already filled with inactive arc. Consequently the vacant node of last position number is over-written by the procedure 2 operation.

4.3 Process of over-writing operation

Taking the operation in Fig.12(b) as an example, process of over-writing operation is described. Fig.13(a) shows operation of procedure 2 over-writing the portion composed of position number and requiered node with the inactive arc. Fig.13(b) illustrates the process of the operation.

First, the string which stands for the active arc and which will be over-written by the inactive arc is introduced as string s0. From string s0 the over-written string composed of position number and requiered node is eliminated. This elimination breaks the string s0 into forepart string s1 and hinder part string s2. The string of the inactive arc is s3. Here an end of letter string is expressed by 'Yen mark plus 0'.

Consequently, inactive arc string s3 is connected to the end of s1. The result of the connection is newly named as s1. Then string s2 is connected to the end of new s1. This result is named s1 again. The finally obtained s1 is the over-writing result.

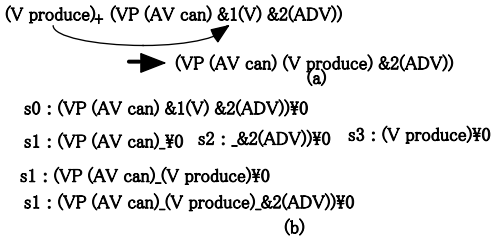

Fig.13 Process of over-writing operation

Strings s0, first s1, s2 and s3 are placed at the second line of Fig.13(b). The string s1 obtained in the course of operation is shown at the third line. The string s1 obtained as the over-write result is placed at the fourth line.

5 Summary

The KATE type parser where procedures 1' , procedure 1 and procedure 2 are conducted entirely in letter string region is realized. This parser can generate trees with OR node which bundles plural partial trees with the same span. This parser has simple structure realizing small line number when coded in C.

References:

[1] M.Kay; Algorithm Schemata and Data Structures in Sytactic Processing; Technical Report CSL-80-12, Xerox PARC(Oct, 1980)

[2]V.R.Pratt; LINGOL-A Progress Report; IJICAI-4 (1975)

[3]H.Tanaka, T.Sato, F.Motoyoshi; Program System for Natural Language Processing--Extended LINGOL(in Japanese);Trans JIECE Vol. J60-D, 12; Dec. 1977

[4]H.Sakaki, K.Hashimoto, M.Suzuki, I.Nogaito, T.Tanaka; A Parsing Method of Natural Language by Filtering Procedure, Trans JIECE, Vol.E-69, 10; Oct 1986

[5]F.Morawiets; Chart Parsing and Constraint Programming, Proc. COLING 2000 pp 551-557

[6] H.Watanabe; A Method for Accelerating CFG-Parsing by Using Dependency Information, Proc. COLING 2000 pp 913-918.

[7]M.Becker, A. Frank; A Stochastic Topological Parser for German, Proc. COLING 2002 pp71-77