A Language Transfer Procedure where Entire Transformation Process is Conducted in the Letter String Region

Hiroshi Sakaki
Faculty of Informatics
Meisei University
2-590 Nagafuchi Ohme-shi 198-8655
Tokyo Japan
sakaki@ei.meisei-u.ac.jp

Takashi Tanaka
SCC Company
5-36-14(No.2 EDC Build.)
Nakano Nakano-ku
164-0001 Tokyo JAPAN

Michihiko Seki
Pro-Log Company
3-14-2 Amanuma Suginami-ku
167-0032 Tokyo JAPAN

Abstract

This paper treats language transfer process in machine translation system. At the process, partial trees in the source language tree are transferred into target language partial trees. Then target language partial trees are merged into the target language tree. Entire data structures processed in the system are letter strings.

Keywords

Machine translation, Language transfer, Partial tree, Transfer by partial tree

1. Forewords

Usually a machine translation system is composed of analysis portion, transfer portion and generation portion. This paper presents a method to compose the transfer portion and generation portion of machine translation system.

There reported two types of intermediate structure. The first type takes the shape of dependency structure or F-structure of LFG typically depicted by the papers by Dorma and Uchida[1],[3]. The second type takes the shape of phrase structural analysis tree typically represented by papers of Sakaki and Watanabe[2][4]. The first type has the advantage that it is suitable for multi-lingual translation. The second type has the characteristics that no processing to obtain intermediate structure is necessary because phrase structural analysis result of source language sentence is already the intermediate structure.

2. Expression of tree structures

A tree structure is expressed by letter string in the way of Fig.1. Fig.1(a) shows universal illustration of the method. The upper part is the tree structure and the lower part is its letter string expression. As seen in Fig.1(a), the letter string consists of a parenthesis pair in which 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.

Each of child structures has its own letter string at which the first and last letters are parenthesis. A child structure composed only of one node is expressed by the name of the node.

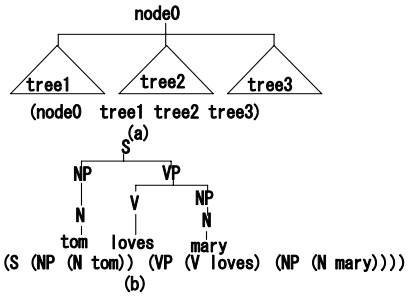


Figure 1 Expression of tree by letter string

Fig.1(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, word nodes are ex
pressed only by lower case letters. In this paper, the explanation for categorical nodes is omitted because the nature of these nodes does not affect the transfer operation.

3. Transfer by partial tree

3.1 Introduction to transfer by partial tree

Operate function (1)**layer(source language data tree)**.
Operate function (2)**transfer(sum)**.
Operate function (3)**buildtree**.
Operate function (4)**generation**.

Figure 2 Operation of process of transfer by partial tree

Fig.2 is the operation of **process of transfer by partial tree.** Fig.3 defines the operation of function **layer**. Fig.4 and Fig.5 define the operation of function **match** and function **transfer** respectively.

Though precise algorithm of this system is stated in Figs. 3, 4, 5, 8, 11 and 15, readers can obtain schema of system by tracing the treatment of examples shown in sections 3.2 and 3.5.

As seen in Fig.3, function **layer** calls itself recurrently at operation (1-3). Function **match** called at operation (1-1) conducts substantial operation of function **layer**. The effect of function **match** and function **transfer** is explained at Section 3.2. The function **buildtree** is treated at Section3.6 and function **generation** is treated at Section 3.7. The explanation of function **layer** is postponed to Section 3.5.

Function **layer** takes **source language data tree** that is a partial tree of **source language analysis tree**, as the argument. Hereafter, a digit in parenthesis affixed to each function designates the function number.

(1)**function layer(source language partial data tree)**
(1-1)Operate function (5)**match(source language partial data tree, source language model tree)** scanning the register of **transfer rules** including **source language model trees**. If function **match** fails for entire **source language model trees** return failure to upper layer. If function **mach** succeeds for a **source language model tree**, go to operation (1-2).
(1-2)Store united body composed of **source language covering tree** and **target language model tree** to variable **sum**. Variable **sum** is accumulation portion of the body. This body has been composed at function **match** in operation (1-1).
(1-3)If no **lower source language partial tree** exists go to operation (1-4). Otherwise recurrently operate function (1)**layer(lower source language partial tree)** for entire **lower source language partial data trees**. If function **layer** fails for one of **lower source language partial data trees**, return failure to upper layer. If function **layer** succeeds for entire **lower source language partial trees**, go to term (1-4).
(1-4)Return success to upper layer.

Figure 3 Operation of function layer

(5)**function match(source language partial data tree, source language model tree)**
(5-1) **Top-cover source language partial data tree** with **source language model tree** included in **transfer rule**. If the **top-covering** is successful go to term (5-2). If not return with failure.
(5-2)Create united body composed of **source language covering tree** and **target language model tree** and set the **transfer rule** as its initial value.
(5-3)Carry over the topmost node number of **source language partial data tree** to the topmost node number of **source language covering tree** in the united body.
(5-4)Generate node numbers at the leaf nodes of **source language covering tree** and the nodes of **source language partial data tree** covered by the leaf node. If a leaf node of **source language partial data tree** coincides with a leaf node of **source language covering tree**, no node number is generated. This operation concludes generation of **source language covering tree**.
(5-5)The portions not covered by **source language covering tree** are designated as **lower source language partial data trees**.

Figure 4 Operation of function match

(2)**function transfer(sum)**
(2-1)Introduce a united body composed of **source language covering tree** and **target language model tree** existing as an element of **sum**.
(2-2)Copy each node number of source language covering tree to the node of target language model tree having the same inside number. This operation generates target language covering tree.

Figure 5 Operation of function transfer

3.2 Sketch of transfer by partial tree

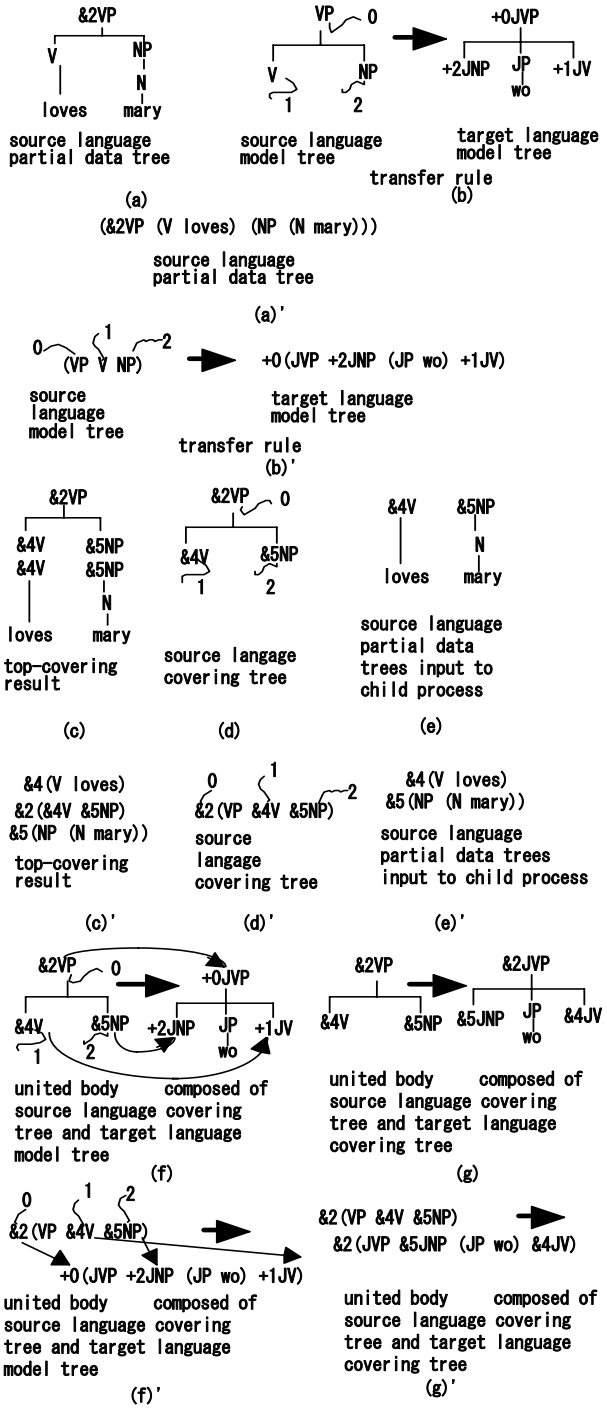


Figure 6 Configurations treated at function match and function transfer

First, the definition of t**op-covering** is necessary. **top-covering** is the process where the partial tree of the **source language partial data tree** including the topmost node is declared as the same as the entire construction of **source language model tree**.

Through the treatment of a simple example shown in Fig.6, principal operation of process of **transfer by partial tree** is sketched. In Fig.6, sub-figures with names without primes show operations in tree structural description. Sub-figures with names attached with primes show operations in letter string region.

Fig.6(a) is **source language partial data tree** input to function **match**. The node number &2 is already given to the topmost node of the tree. A node number is defined to be a sequence of designation letter "&" and a digit. Node numbers are prefixed to the category names. Fig.6(b) is a **transfer rule**. This shows that the **source language model tree** to the left of the arrow is transferred to the **target language model tree** situated to the right of the arrow.

Fig.6(c) is the result generated by **top-covering** the **source language partial data tree** of Fig.6(a) with the **source language model tree** in Fig.6(b). Input **source language partial data tree** is divided into two portions one of which is **top-covered** by **source language covering tree** and the other of which exists out of covering range. The border nodes exist in duplicated fashion.

The initial value of **source language covering tree** is **source language model tree** that is a portion of transfer rule and is obtained at operation (5-2). At the operation (5-4), node numbers &4 and &5 at the leaf positions are newly given automatically by the system, whereas the node number &2 at the topmost node is carried over from the **source language partial data tree** through the operation (5-3). The **source language covering tree** thus generated is shown in Fig.6(d). The **source language covering tree** and **target language model tree** constitute a united body.

The united body of **source language covering tree** and **target language model tree** is added to the variable **sum** by operation (1-2) of function **layer**. Function **transfer** picks up the element of contents in variable **sum** later.

The portions existing below of covering range by **source language covering tree** are generated by operation (5-5) and becomes the **lower source language partial data trees** input to function **match** of child layer as **source language partial data trees** of the child layer. This portion is shown in Fig.6(e). The topmost nodes at each of child **source language partial data trees** in Fig.6(e) have node numbers. As seen in the **source language partial data tree** in Fig.6(a), the node number at the topmost node is given by the parent stage.

The following is illustration of the function **transfer**. The input to this function is a united body composed of **source language covering tree** and **target language model tree** shown in Fig.6(f). The introduction is made by operation (2-1). This is an element in variable **sum** generated by function **match**. To the topmost node and leaf nodes of a **source language covering tree**, **inside numbers** are attached. The numbers are designated using pulled out lines in Fig.6(b) and Fig.6(f). These are so designated because these numbers are not explicitly given. To the topmost node, number 0 is given. Numbers at the leaf nodes increase by the order of left to right.

To each node in **target language model tree** in Fig.6(f), a **inside number** is given after the designation letter +. Each node in the **target language model tree** is the transfer result of the node of the same **inside number** in the **source language covering tree**. In the case of Fig.6(f), the change of relative positions among leaf nodes is observed.

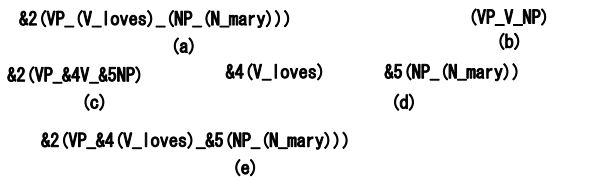
To the node **wo** in Fig.6(f), no **inside number** is attached because this has no corresponding node at the **source language covering tree**.

By the operation (2-2), the node number attached to each node in the **source language covering tree** is transferred to the node in **target language covering tree** having the same **inside number**. The transfer is illustrated in Fig.6(f).

This generates the united body composed of **source language covering tree** and **target language covering tree** shown in Fig.6(g).

3.3 Operation of function match in letter string region

This section deals with the operation of function **match**



in letter string region. Here, formerly used example is used.

Fig.7 Letter strings treated at function match

In Fig.7, Fig.6(a)', Fig.6(b)', Fig.6(d)' and Fig.6(e)' are respectively reproduced in Fig.7(a), (b), (c) and (d). For the purpose of clarification, a space is expressed by a letter "_".

As the letter string in Fig.7(e) has the information of 2 letter strings in Fig.7(d), string in Fig.7(e) replaces that of Fig.7(d). The information of Fig.7(d) is obtained from letter string of Fig.7(e) by selection of sub-strings each of which conforms to the condition that it begins with a sequence composed of the letter & and a digit and that it has the same number of open and close parenthesis.

Fig.8 is the operation of function **match** in letter string region. Letter "s" is attached to each operation number to designate that it treats letter string. In Fig.8, letter strings composing **source language partial data tree** are abbreviated as **data**. Letter strings composing **source language model tree** or **source language covering tree** are abbreviated as **model**. The term **word** means a sequence of letters limited by the letters "(" or ")" or "_" at both ends.

Here the operation in Fig.8 is applied to the input of Fig.7(a) which is **data** and Fig.7(b) which contains **model**. Fig.9 shows the process. In Fig.9, lines with arrows at both

ends show the reading position. Double underlines are used to clarify the reading positions.

(s-5)**function match(source language partial data tree, source language model tree)**
(s-5-1)At the commencement of reading **data**, skip the sequence composed of letter & and a digit.
(s-5-2) Go to operation (s-5-6) when exhaust of string happens at both of **data** and **model**. If not go to next operation.
(s-5-3)If the word beginning at present reading position in **data** is identical with the word beginning at present reading position in **model**, proceed to the position one letter after the word end at both of **data** and **model** and go to operation (s-5-2). If not go to next operation.
(s-5-4)If the letters at the reading positions of **data** and **model** are identical and are "(" or ")" or "_", proceed reading as long as the situation lasts. When the letters at the reading position of **data** and **mode** are different after the break of the situation, go to next operation. When they are identical after the break of situation, go to operation (s-5-2).
(s-5-5)If the letter at the reading position in "**data**" is "(" and the word beginning at the position next to the reading position is identical to the word beginning at the reading position in **model**, attach node numbers to words in either of **data** and **model**. In addition, proceed to the position one letter after the word end at **model**. Moreover, proceed to the position one letter after the position where number of open and close parentheses balances. Then go to operation (s-5-2). If above condition is not met go to the operation (s-5-7).
(s-5-6)Operation of function match successfully ends.
(s-5-7)Operation of function match fails. This situation means that no top-covering tree is detected.

Figure 8 Operation of function match in letter string region

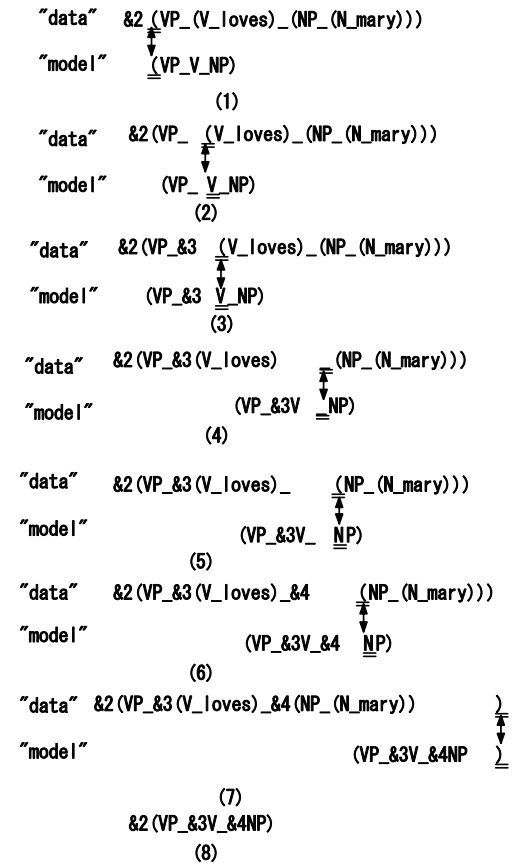
Fig.9(1) shows the reading positions after the application of operation (s-5-1). Fig.9(2) is the reading positions after the effect of application of operations (s-5-3) and (s-5-4). Here, the letter of reading position in **data** is "(" and the word beginning at one letter after reading position in **data** is V. This word is identical with the word beginning at the reading position in **model**. This triggers operation (s-5-5).

First, automatically generated node numbers "&3" are attached to either of **data** and **model** causing situation of Fig.9(3). The reading positions remain at the same position here. By the instruction of further statement in (s-5-5), reading positions move to the positions shown in Fig.9(4).

By the effect of operation (s-5-4) reading positions at both of **data** and **model** proceed by 1 letter reaching the situation in Fig.9(5). First half of operation (s-5-5) generates node number "&4" at either of **data** and **model** giving rise to the situation in Fig.9(6). The later half statements of operation (s-5-5) moves reading positions to the position of Fig.9(7).

Then operation (s-5-4) moves reading positions by one letter at both strings. This leads to the exhaust of strings and the **top-covering** successfully terminates going through operations (s-5-2) and (s-5-6).

The result of operation so far illustrated brings about two strings shown in Fig.9(7). The upper portion is **lower source language partial data trees** of Fig.9(7) fed for function **match** in child layers. The upper portion is



identical to Fig.7(e).

Fig.9 Letter strings treated at the operation of function match

To lower portion of Fig.9(7) the node number of **source language partial data tree** existing in Fig.9(1) is attached giving rise to the letter string in Fig.9(8). This is the **source language covering tree** shown in Fig.7(c).

3.4 Operation of function transfer in letter string region

Fig.10(b) shows a united body composed of **source language covering tree** and **target language covering tree**. This is duplication of Fig.6(g) which is output of function **transfer** Fig.10(a)' is letter string expression of Fig.10(a) that has the same information as Fig.6(f)'. The letter "#" stands for an arrow in Fig.6 and designates the transformation where the portion left of # is source language covering tree and the portion to the right of # is target language model tree. The letter "|" designates the end of a united body.

This operation is illustrated by arrows in Fig.10(a)'. Fig.10(b)' is letter string expression of the united body composed of **source language covering tree** and **target**

language covering tree. Fig.10(b)' is identical with construction of Fig.10(b) or Fig.6(g).

Fig.11 is the operation of function **transfer** in letter string region. In operation in Fig.11, parameter r represent **inside number** common to **source language covering tree** and **target language model tree**.

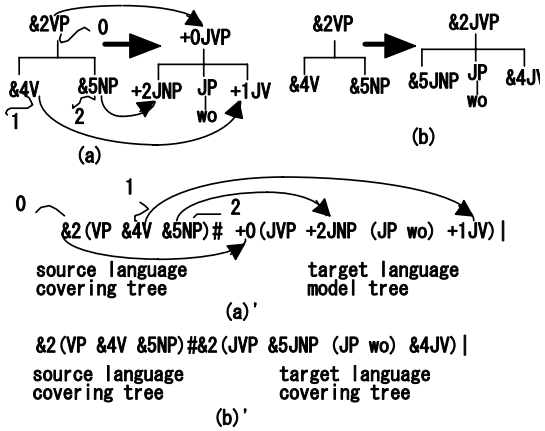


Figure 10 Processing at the function transfer

(s-2)**transfer(sum)**
(s-2-1) Introduce united body composed of **source language covering tree** and **target language model tree** from variable **sum**.
(s-2-2) Set initial value of **inside number** r to 0. Then go to next operation (s-2-3). If r'th '&' does not exist in the string of **source language covering tree** situated to the left of letter #, go to operation (s-2-7). Otherwise, go to next operation.
(s-2-4) Scan **source language covering tree** string and obtain node number attached to r'th '&'.
(s-2-5) Replace **inside number** r situated after the letter '+' at **target language model tree** string with the node number obtained by operation (s-2-3). Then go to next operation.
(s-2-6) Increase value of r by 1. Then go to (s-2-3) operation.
(s-2-7) Terminate transfer operation.

Fig.11 operation of function transfer in letter string region

3.5 Application of the process of the transfer by the partial tree

In this section, the operation of process of **transfer by partial tree** given in section 3.1 is applied to an **analysis tree** representing whole sentence. As transfer operation is possible by process in letter string region, only tree structural expression is treated here.

The example treated here deals with English sentence "tom loves mary" that includes former example as its local structure. Fig.12 traces the transfer process for the example.

The first digits in the name of sub-figures represent respective application layer of the function **layer** introduced in Section 3.1. Sub-figures having digits 1, 2, 3, 4 at the second positions in their names respectively contains **source language partial data trees**, **transfer rules**, **top-covering results** and **target language covering tree**.

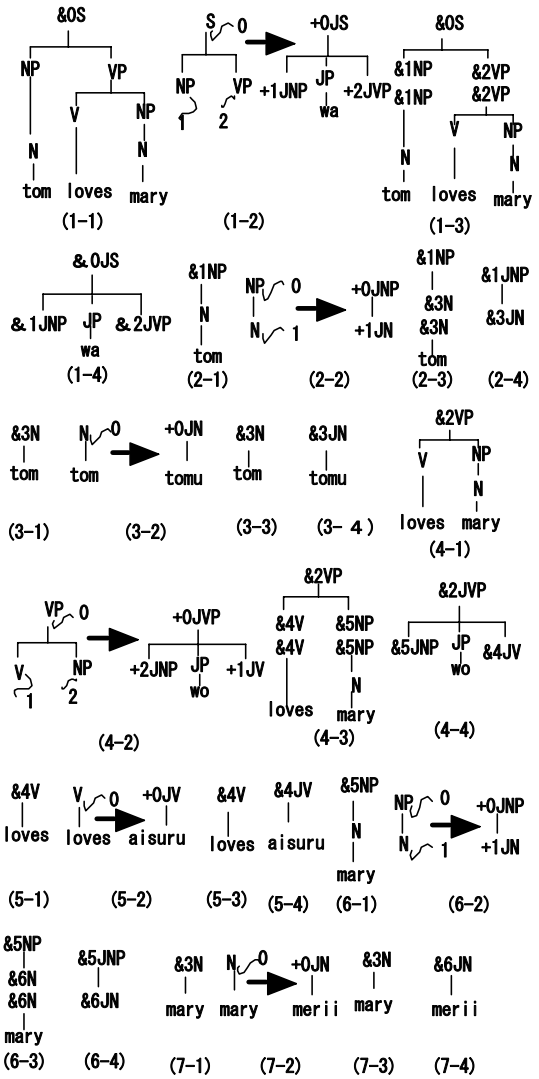


Figure 12 Treatment of analysis tree of a sentence

At the application of function **layer** to the root layer depicted in Fig.3, the tree of Fig.12(1-1) is input as **source language data tree**. By the operation (1-1) function **match** succeeds using **transfer rule** of Fig.12(1-2). Function **match** generates **top-covering** situation in Fig.12(1-3) generating **source language covering tree** and **lower source language partial data trees**. Among them **source language covering trees** are accumulated in the variable **sum** by the operation (1-2).

At the operation (1-3) 2 child layers are created and the function **layer** is called twice respectively with the input of **source language partial trees** of Fig.12(2-1) and Fig.12(4-1). As these child layers return success, function **layer** at root layer become successful.

Investigation is made for the layer input with the **lower source language partial data tree** of Fig.12(2-1). This layer succeeds using **transfer rule** of Fig.12 (2-2). It generates **top-covering** situation of Fig.12(2-3). As the child layer called at operation (1-3) is successful, this layer become successful.

Then, investigation is made for child layer of the lastly treated layer. The input is Fig.12(3-1). This layer succeeds using **transfer rule** of Fig.12(3-2). It generates **top-covering** situation of Fig.12(3-3). This layer succeeds because no child layer exists.

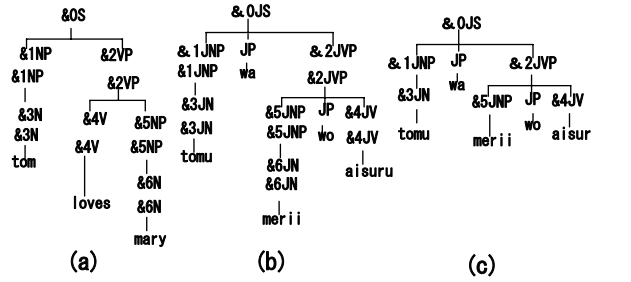


Figure 13 Covering trees and target language tree

Fig.13(a) shows **source language covering trees** existing in Fig.12. At Fig.13(a), identical nodes are placed near to each other. Fig.13(b) shows **target language covering trees** arranged similarly.

Incorporation of identical nodes in Fig.13(b) will bring about **target language tree** aimed at. The result of incorporation is shown in Fig.13(c).

3.6 Operation of function buildtree in letter string region

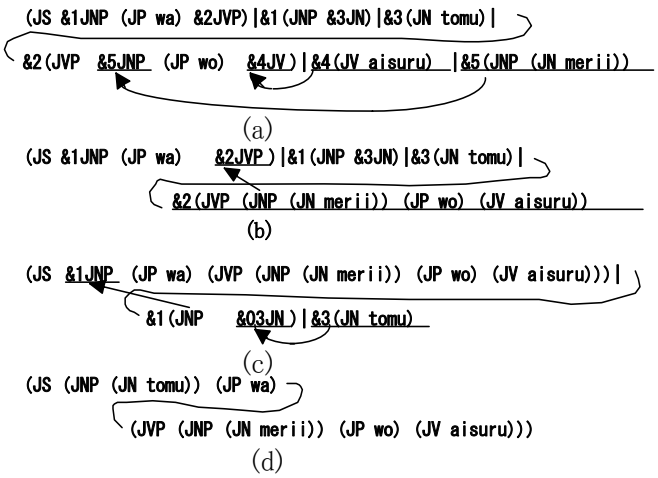


Figure 14 Letter strings treated at the Operation of function buildtree

This section deals with the operation of function **buildtree** in letter string region. This function takes the input of **target language covering trees** shown in Fig.13(b) and generates **target language tree** in Fig.13(c).Fig.14 is the figure of letter strings generated at operation of function **buildtree** working in letter string region. Fig.15 is the operation of function **buildtree** in letter string region.

(3)**buildtree**
(s-3-1) Search node number from the rear of the **target language covering tree** aggregation and obtain the first node number encountered together with the structure following the number. If the node number is absent go to operation (s-3-3) else go to next operation.
(s-3-2) Search node number identical with the number obtained at operation (s-3-1) from the front of the aggregation. Replace the node number and the node attached to the number with the structure obtained at operation (s-3-1).
(s-3-3) Terminate the tree connection operation.

Fig.15 Operation of function buildtree in letter string region

The first line in Fig.14 is **target language covering tree** aggregation obtained at the completion of function **transfer**. By the consequence of operations (s-3-1) and (s-3-2) the portions of **target language covering tree** having node number &5 and &4 respectively replaces nodes with node numbers &5 and &4. Continuing the operation the line at the bottom is obtained. This is the **target language tree** of Fig.13(c) which is the transfer result of the **source language tree** in Fig.12(1-1).

4 Conclusion

A language transfer system utilizing **transfer by partial tree** method and conducting entire operation in the letter string region is built. This system uses phrase structural tree as intermediate structure. As the large-scale system called KATE utilizing **transfer by partial tree** can translate complex sentences, this system will be also capable of treating such sentences[2]. The operation of KATE system is conducted in list structural region. The direction of translation aimed at by the system of this paper is solely English to Japanese translation.

A C language program of 300 lines realizing **transfer by partial tree** is build. This relatively small line number is the effect of treatment in letter string region.

The authors of this paper have also built a parser where entire operation is conducted in letter string region. The result is scheduled to appear in the proceedings of 7th WSEAS Conference on Applied Informatics and Communication (AIC'07)

References

[1]M. Dorna, A. Frank, J. van Geneabith, M. Emele ; *Syntactic and Semantic Transfer with F-Structure*; Proc. COLING 98 pp. 341-347; Aug. 1998

[2]H.Sakaki, K. Matsumoto, S. Kuroiwa, K. Hashimoto; *Natural Language Transfer System by Recurring Algorithm*(in Japanese); Trans. JIECE Vol. J72-D-II No.12 pp.2080-2093 ; Dec. 1989

[3]H. Uchida, M. Zhu, T. Della Senta ; *UNL Universal Networking Language*, UNDL Foundation 2005

[4]H. Watanabe, K. Takeda; *A Pattern-based Machine Translation System Extended by Example-based Processing*; Proc. COLING 98 pp. 1396-1373; Aug. 1998