Integration of Morphological and Syntactic Analysis based on LR Parsing Algorithm

Tanaka Hozumi,[†] Tokunaga Takenobu[†] and Aizawa Michio[††]

Morphological analysis of Japanese is very different from that of English, because no spaces are placed between words. This is also the case in many Asian languages such as Korean, Chinese, Thai and so forth. In the Indo-European family, some languages such as German have the same phenomena in forming complex noun phrases. Processing such languages requires the identification of the word boundaries in the first place. This process is often called *segmentation*. Segmentation is a very important process, since the wrong segmentation causes fatal errors in the later stages such as syntactic, semantic and contextual analysis. However, correct segmentation is not always possible only with morphological information. Syntactic, semantic and contextual information are also necessary to resolve the ambiguities in segmentation. This paper proposes a method to integrate the morphological and syntactic analysis based on LR parsing algorithm. An LR table derived from grammar rules is modified on the basis of connectabilities between two adjacent words. The modified LR table reflects both the morphological and syntactic constraints. Using the LR table and the generalized LR parsing algorithm, efficient morphological and syntactic analysis is available.

KeyWords: *Generalized LR parsing, Morphological analysis, Syntactic analysis*

1 Introduction

Morphological analysis of Japanese is very different from that of English, because no spaces are placed between words. This is also the case in many Asian languages such as Korean, Chinese, Thai and so forth. In the Indo-European family, some languages such as German have the same phenomena in forming complex noun phrases. Processing such languages requires the identification of the word boundaries in the first place. This process is often called *segmentation*. Segmentation is one of the most important tasks of morphological analysis for these languages, since the wrong segmentation causes fatal errors in the later stages such as syntactic, semantic and contextual analysis. However, correct segmentation is not always possible only with morphological information. Syntactic, semantic and contextual information are also necessary to resolve the ambiguities in segmentation.

Over the past decades a number of studies have been made on the morphological and

† Department of Computer Science, Tokyo Institute of Technology
†† Media Technology Laboratory, Canon Inc.

syntactic analysis of Japanese. From the viewpoint of two kinds of integration, description of constraints and processing, they would be classified into the following three approaches. The relation of these approaches is illustrated in Figure 1.

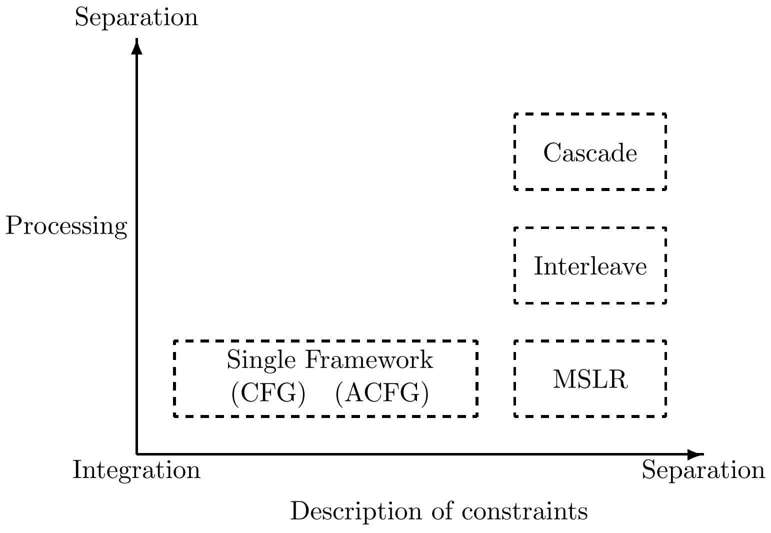


Figure 1 Relation of approaches

Cascade: Separate the morphological and syntactic analysis and execute them in a cascade manner. The morphological and syntactic constraints are represented separately.

Interleave: Separate the morphological and syntactic analysis and execute them interleavingly. The morphological and syntactic constraints are represented separately.

Single Framework: Represent both the morphological and syntactic constraints in a single framework such as context free grammars (CFGs) and make no distinction between the two analysis.

Representing the morphological and syntactic constraints separately as in the first two approaches, Cascade and Interleave, makes maintaining and extending the constraints easier. This is an advantage of these approaches. Many natural language processing systems have used these two approaches. For example, Mine et al. proposed a method to represent the morphological constraints in regular grammar and the syntactic constraints in CFG, and interleave the morphological and syntactic analysis (Mine, Taniguchi, and Amamiya 1991). Most other systems use a connection matrix instead of regular grammar (Miyazaki 1984; Sugimura, Akasaka, Kubo, and Matsumoto 1988). The main drawbacks of these approaches would be summarized as follows:

- It may require two different algorithms for each analysis.
- It must retain all ambiguities from the morphological analysis until the syntactic anal-

ysis begins. This wastes memory space and computing time.

On the other hand, from a viewpoint of processing, it is preferable to integrate the morphological and syntactic analysis into a single framework, since some syntactic constraints are useful for morphological analysis and vice versa. The last approach fulfills this requirement. There have been several attempts to develop CFG that covers both the morphological and syntactic constraints (Kita 1992; Sano and Fukumoto 1992). However, it is empirically difficult to describe both constraints by using only CFG. In order to have CFG rules include morphological constraints, nonterminal symbols have to bear the morphological attributes which are used for checking connectabilities between morphemes. In other words, nonterminals should be more precisely subcategorized. This increases the number of nonterminals and thus that of grammar rules.

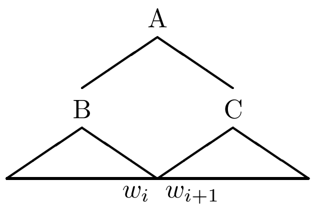


Figure 2 Connectability check by CFG

Using augmented context free grammar (ACFG) instead of CFG may remedy this problem. However, this may cause the delay of connectability checking. For example, in Figure 2, in order to check the connectability between adjacent words, w_i and w_{i+1}, the morphological attributes of each word should be propagated up to their mother nodes B and C, and the check is delayed until the application of the rule A $\rightarrow$ B C.

By using connection matrices for morphological analysis as in the Cascade/Interleave approaches, connectability checks between adjacent words is performed very easily. Therefore, it is desirable to represent the morphological and syntactic constraints separately as in Cascade/Interleave, and to integrate the execution of both analysis into a single process as in Single Framework. Our method has captured these advantages by representing the morphological constraints in connection matrices and the syntactic constraints in CFGs, then compiling both constraints into an LR table (Aho, Ravi, and Ullman 1986). The already existing efficient LR parsing algorithms would be used with minor modifications, enabling us to utilize both the morphological and syntactic constraints at the same time. Our approach, called MSLR (Morpho-Syntactic LR), locates at the right bottom of the map in Figure 1, which is the most preferable position.

In the next section, we first give a brief introduction to Japanese morphological analysis using an example sentence. In section 3, we describe the method of generating an LR table from a connection matrix and CFG rules, then in section 4 we explain the parsing algorithm. Our algorithm is principally the same as Tomita's generalized LR parsing algorithm (Tomita 1986). The only difference is that the input is not a sequence of preterminals, but a sequence of characters.

2 Morphological analysis of Japanese

A simple Japanese sentence consists of a sequence of postpositional phrases (PPs) followed by a predicate. The PP consists of a noun phrase (NP) followed by a postposition which indicates the case role of the NP. The predicate consists of a verb or an adjective, optionally followed by a sequence of auxiliary verbs (Morioka 1987)).

We illustrate the Japanese morphological analysis with an example sentence "かおるにあいます (meet Kaoru)." We use a simple Japanese dictionary shown in Figure 3, and a connection matrix shown in Figure 4 which gives us the connectabilities between adjacent morphological categories (mcat). For example in Figure 4, the symbol "`o`" at the intersection of row 2 (p_1) and column 3 (vs_k) indicates that the morphological category vs_k can immediately follow the morphological category p_1.

entry	cat	mcat	meaning
かお	n	n_1	face
かおる	n	n_1	person's name
に	p	p_1	(dative)
あ	vs	vs_k	open
き	ve	ve_k^2	(connect to verb)
い	ve	ve_k^{2i}	(connect to verb)
く	ve	ve_k^3	(connect to nominal)
あ	vs	vs_w	meet
い	ve	ve_w^2	(connect to verb)
っ	ve	ve_w^{2t}	(connect to verb)
く	ve	ve_w^3	(connect to nominal)
かお	vs	vs_r	smell sweet
り	vs	ve_r^2	(connect to verb)
っ	ve	ve_r^{2t}	(connect to verb)
る	ve	ve_r^3	(connect to nominal)
ます	ax	ax_1	(polite form)
た	ax	ax_2	(past form)

n	noun
p	case marker
vs	verb stem
ve	verb ending
ax	auxiliary verb

Figure 3 An example of Japanese dictionary

R I G H T

		n_1	p_1	vs_k	vs_r	vs_w	ve_k^2	ve_k^{2i}	ve_k^3	ve_w^2	ve_w^{2t}	ve_w^3	ve_r^2	ve_r^{2t}	ve_r^3	ax_1	ax_2	$
	n_1		o															
	p_1			o	o	o												
	vs_k						o	o	o									
	vs_w									o	o	o						
	vs_r												o	o	o			
L	ve_k^2															o		
E	ve_k^{2i}																o	
F	ve_k^3	o																o
T	ve_w^2															o		
	ve_w^{2t}																o	
	ve_w^3	o																o
	ve_r^2															o		
	ve_r^{2t}																o	
	ve_r^3	o																o
	ax_1																	o
	ax_2																	o

Figure 4 An example of a connection matrix

Using only the dictionary, we would obtain the following twelve candidates of segmentation for the sentence "かおるにあいます．"

	かお	る	に	あ	い	ます
(1)	n_1	ve_r^3	p_1	vs_k	ve_k^{2i}	ax_1
(2)	n_1	ve_r^3	p_1	vs_k	ve_w^2	ax_1
(3)	n_1	ve_r^3	p_1	vs_w	ve_k^{2i}	ax_1
(4)	n_1	ve_r^3	p_1	vs_w	ve_w^2	ax_1
(5)	vs_r	ve_r^3	p_1	vs_k	ve_k^{2i}	ax_1
(6)	vs_r	ve_r^3	p_1	vs_k	ve_w^2	ax_1
(7)	vs_r	ve_r^3	p_1	vs_w	ve_k^{2i}	ax_1
(8)	vs_r	ve_r^3	p_1	vs_w	ve_w^2	ax_1

	かおる	に	あ	い	ます
(9)	n_1	p_1	vs_k	ve_k^{2i}	ax_1
(10)	n_1	p_1	vs_k	ve_w^2	ax_1
(11)	n_1	p_1	vs_w	ve_k^{2i}	ax_1
(12)	n_1	p_1	vs_w	ve_w^2	ax_1

By also referring to the connection matrix, we would be able to filter out illegal segmentations. From the examples above, we find (1)–(4) violate the connectability between "かお (n_1)" and "る (ve_r^3)", and that (5)–(8) violate the connectability between " る (ve_r^3)" and "に (p_1)." Also (9) and (11) violate the connectability between "い (ve_k^{2i})" and "ます (ax_1)", and (11) violates the connectability between "あ (vs_w)" and "い (ve_k^{2i})." Thus by eliminating the illegal candidates we obtain the morphologically correct candidate, (12). A long input sentence generally gives ambiguities which need to be resolved in later stages using syntactic, semantic and contextual information.

3 Generating LR table

Connection matrices and CFG rules have been used for morphological analysis and syntactic analysis respectively by most Japanese processing systems. Because CFG rules were mainly used for syntactic analysis and connection matrices for morphological analysis, they have been developed independently of each other.

In this section, we propose a method to integrate morphological and syntactic constraints in the framework of LR parsing algorithm, and thus capturing the advantages of Cascade/Interleave and Single Framework described in section 1.

In order to combine connection matrices and CFG rules, the first step we have to take is to extend the CFG rules by relating the syntactic categories in the CFG rules with the morphological categories in a connection matrix. This is realized by adding CFG rules called morphological rules each of which is a unit production rule with a syntactic category in the LHS and a morphological category in the RHS.

$\texttt{s} \rightarrow \texttt{v}$	(1)	$\texttt{v} \rightarrow \texttt{vs ve}$	(4)
$\texttt{s} \rightarrow \texttt{v ax}$	(2)	$\texttt{pp} \rightarrow \texttt{n p}$	(5)
$\texttt{s} \rightarrow \texttt{pp s}$	(3)		

Figure 5 An example of CFG for Japanese

$\texttt{n} \rightarrow n_1$	(6)	$\texttt{ve} \rightarrow ve_w^2$	(14)
$\texttt{p} \rightarrow p_1$	(7)	$\texttt{ve} \rightarrow ve_w^{2t}$	(15)
$\texttt{vs} \rightarrow vs_k$	(8)	$\texttt{ve} \rightarrow ve_w^3$	(16)
$\texttt{vs} \rightarrow vs_w$	(9)	$\texttt{ve} \rightarrow ve_r^2$	(17)
$\texttt{vs} \rightarrow vs_r$	(10)	$\texttt{ve} \rightarrow ve_r^{2t}$	(18)
$\texttt{ve} \rightarrow ve_k^2$	(11)	$\texttt{ve} \rightarrow ve_r^3$	(19)
$\texttt{ve} \rightarrow ve_k^{2i}$	(12)	$\texttt{ax} \rightarrow ax_1$	(20)
$\texttt{ve} \rightarrow ve_k^3$	(13)	$\texttt{ax} \rightarrow ax_2$	(21)

Figure 6 A morphological rules derived from the dictionary in Figure 3

Let us take an exmaple CFG rules shown in Figure 5. From the dictionary shown in Figure 3, we would extract a set of new CFG rules as shown in Figure 6, which are simply added to the CFG rules in Figure 5 to get an extended set of CFG rules.

We would generate an LR table as shown in Figure 7 from the extended CFG rules (1) through (21) from Figure 5 and 6. Note that the extended CFG rules do not include any information about connectability represented in the connection matrix in Figure 4. For example, rules (4), (10) and (14) allow the structure "$\texttt{[v [vs } vs_r\texttt{],[ve [}ve_w^2\texttt{]]]}$" which violates the connectability between vs_r and ve_w^2 with respect to Figure 4.

A C T I O N

state	n_1	vs_k	vs_w	vs_r	ax_1	ax_2	ve_k^2	ve_k^{2i}	ve_k^3	ve_w^2	ve_w^{2t}	ve_w^3	ve_r^2	ve_r^{2t}	ve_r^3	p_1	$
0	s6	s7	s8	s9													
1																	acc
2					s11	s12											r1
3	s6	s7	s8	s9													
4							s15	s16	s17	s18	s19	s20	s21	s22	s23		
5																s25	
6																r6	
7							r8	r8	r8	$\boxed{r8}$	$\boxed{r8}$	$\boxed{r8}$	$\boxed{r8}$	$\boxed{r8}$	$\boxed{r8}$		
8							$\boxed{r9}$	$\boxed{r9}$	$\boxed{r9}$	r9	r9	r9	$\boxed{r9}$	$\boxed{r9}$	$\boxed{r9}$		
9							$\boxed{r10}$	$\boxed{r10}$	$\boxed{r10}$	$\boxed{r10}$	$\boxed{r10}$	$\boxed{r10}$	r10	r10	r10		
10																	r2
11																	r20
12																	r21
13																	r3
14					r4	r4											r4
15					r11	$\boxed{r11}$											$\boxed{r11}$
16					$\boxed{r12}$	r12											$\boxed{r12}$
17					$\boxed{r13}$	$\boxed{r13}$											r13
18					r14	$\boxed{r14}$											$\boxed{r14}$
19					$\boxed{r15}$	r15											$\boxed{r15}$
20					$\boxed{r16}$	$\boxed{r16}$											r16
21					r17	$\boxed{r17}$											$\boxed{r17}$
22					$\boxed{r18}$	r18											$\boxed{r18}$
23					$\boxed{r19}$	$\boxed{r19}$											r19
24	r5	r5	r5	r5													
25	$\boxed{r7}$	r7	r7	r7													

G O T O

state	s	v	pp	vs	n	ax	ve	p
0	1	2	3	4	5			
1								
2						10		
3	13	2	3	4	5			
4							14	
5								24

Figure 7 LR table generated from rules (1)–(21)

The second step is to introduce the constraints on connectability into the LR table by deleting illegal reduce actions. This is carried out by modifying the LR table with the procedure shown in Figure 8.

```
For each reduce action A with a morphological rule in the LR table {
    if (Not $Connect(RHS(Rule(\mathbf{A})), LA(\mathbf{A}))$ {
        delete A from the entry;
    }
}
where each function is defined as follows:
$Rule : action \rightarrow rule$;
    returns a rule used by the reduce action.
$LA : action \rightarrow symbol$;
    returns a look ahead symbols of the action.
$Connect : symbol \times symbol \rightarrow \{T, F\}$;
    returns true or false with respect to the connectability of the two symbols.
$RHS : rule \rightarrow symbol$;
    returns a right hand side symbol of the rule.
```

Figure 8 A procedure to modify an LR table

Deleting reduce actions by applying the above procedure prohibits the application of morphological rules which violates the connectability between two adjacent words, namely the current scanned word and its lookahead word. Note that given an LR table and a connection matrix, this procedure can be performed automatically without human intervention.

It is possible to incorporate this procedure into the LR table generation process, however, it is better to keep them separate. Since this procedure is applicable to any type of LR table, separating this process from LR table generation enables us to use the already existing LR table generation program.

For example, in Figure 7, the reduce action `r8` in state 7 and column ve_r^3 is deleted, since the connection between vs_k, the RHS of rule (8), and ve_r^3, the lookahead symbol, is prohibited with respect to the connection matrix in Figure 4. Similarly, reduce action `r8` in state 7 and column ve_w^2 will be deleted and so forth. The reduce actions which should be deleted is enclosed in Figure 7. The overview of generating a modified LR table is shown in Figure 9.

Generally speaking, the size of the LR table is on the exponential order of the number of rules in the grammar. Introducing the morphological rules into the syntactic rules can cause an increase in the number of states in the LR table, thereby exponentially increasing the size of the overall LR table in the worst case. In our method, the increase of the number of states is equal to that of the morphological rules introduced. Suppose we add a morphological rule $X \rightarrow x$ to the grammar. Only the items in the form of $[A \rightarrow \alpha \cdot X\beta]$ would produce a single new item $[X \rightarrow \cdot x]$ from which only a single new state $\{[X \rightarrow x\cdot]\}$ would be created. Thus the increase of the number of the states is equal to that of the morphological rules introduced, and the size of the LR table will not grow exponentially.

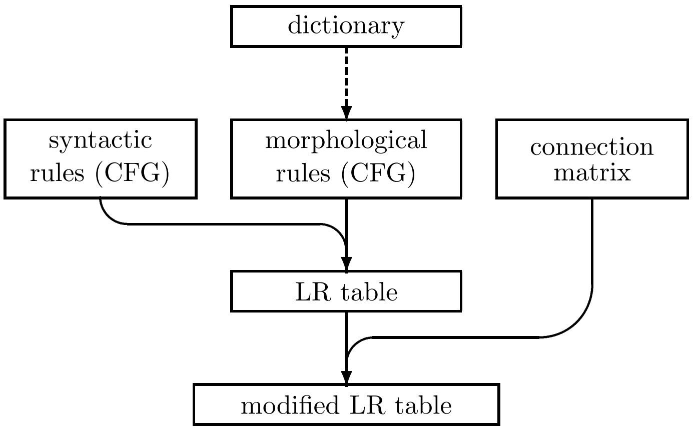


Figure 9 Outline of generating a modified LR table

4 Algorithm

The LR parsing algorithm with the modified LR table is principally the same as Tomita's generalized LR parsing algorithm. The only difference is that Tomita's algorithm assumes a sequence of preterminals as an input, while our algorithm assumes a sequence of characters. Thus the dictionary reference process needs to be slightly modified. Figure 10 illustrates the outline of our parsing algorithm.

In Figure 10 the stage number (CS) indicates how many characters have been processed. The procedure begins at stage 0 and ends at stage N, the character length of an input sentence. In stage 0, the stack is initialized and only the node with state 0 exists (step (1)). In the outer-most loop (2)–(14), each stack top in the current stage is selected and processed. In step (4), the dictionary is consulted and look-ahead symbols are obtained. An important point here is that look-ahead symbols may have different character lengths. A new node is introduced by a shift action at step (8) and is placed into a stage which is ahead of the current stage by the length of the look-ahead word.

```
(1)  initialize stack
(2)  for CS = 0 ... N {
(3)   for each stack top node in stage CS {
(4)     Look-aheads = lookup-dictionary(CS);
(5)     for each look ahead preterminal LA in Look-aheads {
(6)       do reduce while "reduce" is applicable;
(7)       if "shift" is applicable {
(8)         do shift creating a new node in stage (CS + length(LA));
(9)       }
(10)      if "acc" { accept }
(11)      if no action { reject }
(12)    }
(13)  }
(14) }
```

Figure 10 Outline of the parsing algorithm

5 A worked example

The following example well illustrates the algorithm in Figure 10. The input sentence is “かおるにあいます$ ” (meet Kaoru). We assign position numbers between adjacent characters.

Input:		か	お	る	に	あ	い	ま	す	$
Position:	0	1	2	3	4	5	6	7	8	9

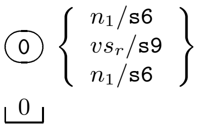
In the following trace, the numbers in circles denote state numbers, and the numbers in squares denote the subtree number. The symbols enclosed by curly brackets denote a look ahead symbol followed by the next applicable action, separated by a slash (/). The stage numbers are shown below the stacks.

Stage: 0

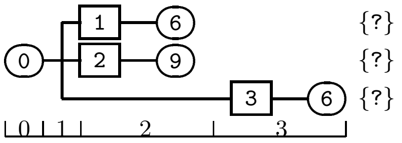
Dictionary reference:

$[n_1$, “かお”]	at 0–2
$[vs_r$, “かお”]	at 0–2
$[n_1$, “かおる”]	at 0–3

We find three look ahead symbols, n_1, vs_r, and n_1 by consulting the dictionary in Figure 3. A shift actions is applied for each of them according to the LR table in Figure 7.



After the shift actions, three new nodes are created at stage 2 or stage 3 depending on the length of look ahead words. At the same time subtrees $\boxed{1}$–$\boxed{3}$ are constructed. The current stage is updated from 0 to 2, since there is no node in stage 1. The look ahead symbols are unknown at this moment.



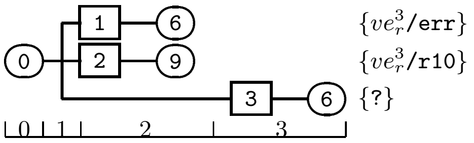
$\boxed{1}$: [n_1, "かお"]
$\boxed{2}$: [vs_r, "かお"]
$\boxed{3}$: [n_1, "かおる"]

Stage: 2

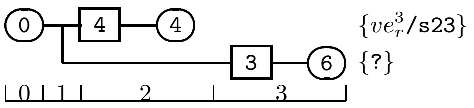
Dictionary reference:

[ve_r^3, "る"] at 2–3

Dictionary reference gives one look ahead symbol from position 2. Since the current stage number is 2, only the first two stack tops are concerned at this stage. No action is taken of the first stack, because the LR table has no action in the entry for state 6 and a look ahead symbol ve_r^3. As the result, the first stack is rejected. The reduce action (r10) is taken for the second stack.

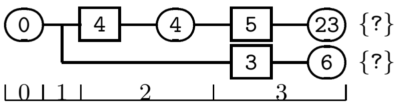


After r10, a shift action (s23) is carried out for the first stack.



$\boxed{4}$: [vs, $\boxed{2}$]

After s23, we would proceed to stage 3.



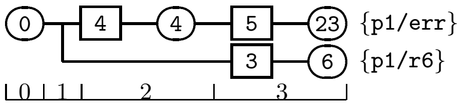
$\boxed{5}$: [ve_r^3, "る"]

Stage: 3

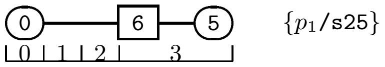
Dictionary reference:

[p_1, "に"] at 3–4

We obtain symbol p_1 by consulting the dictionary. Because the first stack can take no more action, it is rejected. The reduce action (r6) is then applied to the second stack.

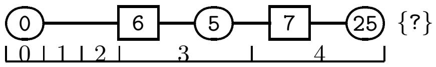


The shift action (s25) is applied to the following stack.



$\boxed{6}$: [n, $\boxed{3}$]

After the shift action (s25), new nodes are created in stage 4.



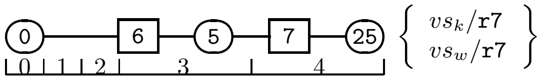
$\boxed{7}$: [p_1, “に”]

Stage: 4

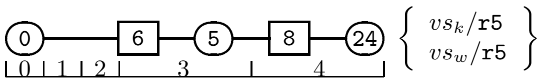
Dictionary reference:

[vs_k, “あ”]	at 4–5
[vs_w, “あ”]	at 4–5

Dictionary reference provides two look ahead symbols for the next word.

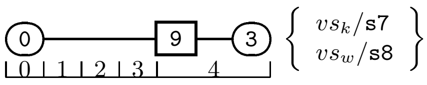


After the two reduce actions (r7), we get two nodes with the same state 24, and they would be merged. This is possible because these two reduce actions give the same structure as well. If the structures are different, we would not able to merge the stacks. We would see such an example later at stage 5.

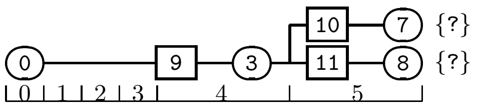


$\boxed{8}$: [p, $\boxed{7}$]

The process in stage 4 continues as follows.



$\boxed{9}$: [pp, $\boxed{6}$, $\boxed{8}$]



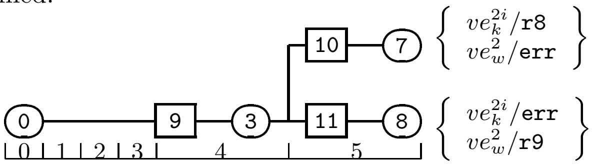
$\boxed{10}$: [vs_k, “あ”]
$\boxed{11}$: [vs_w, “あ”]

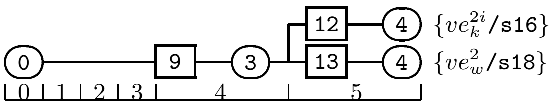
Stage: 5

Dictionary reference:

$[ve_k^{2i}$, “い”$]$	at 5–6
$[ve_w^{2}$, “い”$]$	at 5–6

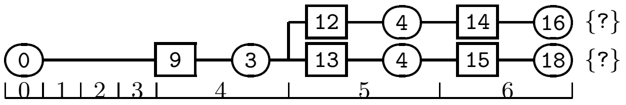
We have two look ahead symbols for each stack top. The reduce actions (`r8` and `r9`) are performed.


{ ve_k^{2i}/r8, ve_w^{2}/err }
{ ve_k^{2i}/err, ve_w^{2}/r9 }



$\boxed{12}$: [vs, $\boxed{10}$]
$\boxed{13}$: [vs, $\boxed{11}$]

Note that we are not able to merge the stack tops even with the same state 4 since the structure of $\boxed{12}$ and $\boxed{13}$ are different. If two stack tops are merged here and then different shift actions (`s16` and `s18`) are carried out, we might have invalid combinations of structure such as ($\boxed{12}$, $\boxed{15}$) and ($\boxed{13}$, $\boxed{14}$).



$\boxed{14}$: $[ve_k^{2i}$, “い”$]$
$\boxed{15}$: $[ve_w^{2}$, “い”$]$

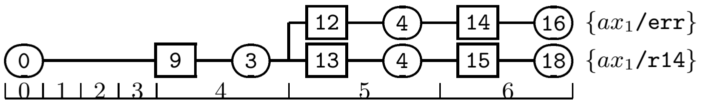
After the shift actions (`s16` and `s18`), we proceed to stage 6.

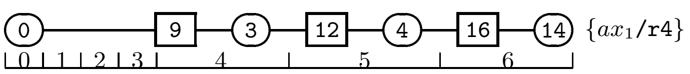
Stage: 6

Dictionary reference:

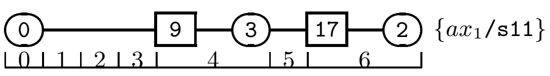
$[ax_1$, “ます”$]$	at 6–8

The process in stage 6 proceeds as follows.

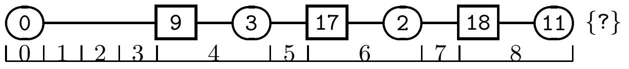




$\boxed{16}$: [ve, $\boxed{15}$]



$\boxed{17}$: [v, $\boxed{12}$, $\boxed{16}$]

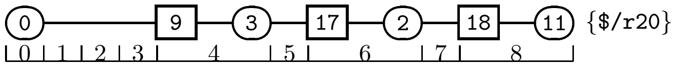


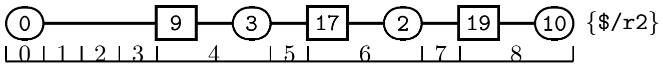
$\boxed{18}$: [ax_1, “ます”]

Stage: 8

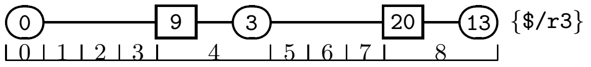
Dictionary reference:

“$” at 8–9



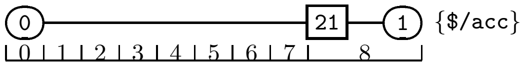


$\boxed{19}$: [ax, $\boxed{18}$]



$\boxed{20}$: [s, $\boxed{17}$, $\boxed{19}$]

The input sentence is automatically segmented and accepted, giving a final parse result 21 as shown in Figure 11.



$\boxed{21}$: [s, $\boxed{9}$, $\boxed{20}$]

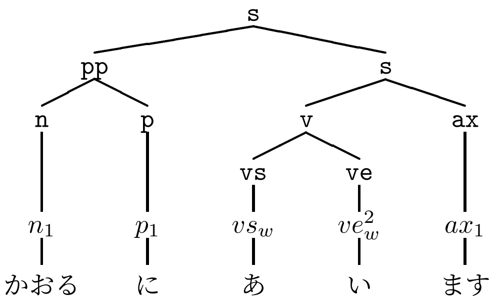


Figure 11 An analysis of “かおるにあいます”

6 Conclusion

We have proposed a method representing the morphological constraints in connection matrices and the syntactic constraints in CFGs, then compiling both constraints into an LR table. The compiled LR table enables us to make use of the already existing, efficient generalized LR parsing algorithms through which integration of both morphological and syntactic analysis is obtained.

Advantages of our approach would be summarized as follows:

- Morphological and syntactic constraints are represented separately, and it makes easier to maintain and extend them.
- The morphological and syntactic constraints are compiled into a uniform representation, an LR table. We can use the already existing efficient algorithms for generalized LR parsing for the analysis.
- Both the morphological and syntactic constraints can be used at the same time during the analysis.

We have implemented our method using the EDR dictionary with 300,000 words (EDR 1993) from which 437 morphological rules are derived. This means only 437 new states are introduced to LR table and this does not cause an explosion in the size of the LR table. The method proposed in this paper is also applicable to integrate phonological and syntactic analysis. The detail is described elsewhere (Tanaka, Li, and Tokunaga 1994).

Reference

Aho, A., Ravi, S., and Ullman, J. (1986). *Compilers, Principle, Techniques, and Tools.* Addision Wesely.

Japan Electronic Dictionary Research Institute (1993). *EDR Dictionary Manual.*

Kita, K. (1992). *A Study on Language Modeling for Speech Recognition.* Ph.D. thesis, Waseda University.

Mine, T., Taniguchi, R., and Amamiya, M. (1991). “Coordinated Morphological and Syntactic Analysis of Japanese Language.” In *Proceedings of IJCAI’91*, pp. 1012–1017.

Miyazaki, M. (1984). “An Automatic Segmentation Method for Compound Words using Dependency Analysis.” *Transactions of Information Processing Society of Japan*, *25*(6), 970–979.

Morioka, K. (1987). *Formation of a Vocabulary.* Meizi-syoin.

Sano, H., and Fukumoto, F. (1992). “On a Grammar Formalism, Knowledge Bases and Tools for Natural Language Processing in Logic Programming.” In *Proceedings of FGCS’92.*

Sugimura, R., Akasaka, K., Kubo, Y., and Matsumoto, Y. (1988). “Logic Based Lexical Analyzer LAX.” In *Logic Programming ’88.* Springer-Verlag.

Tanaka, H., Li, H., and Tokunaga, T. (1994). “Incorporation of Phoneme-Context-Dependence in LR Table through Constraint Propagation Method.” In *Proceedings of the Integration of Natural Language and Speech Processing*, pp. 15–22.

Tomita, M. (1986). *An Efficient Parsing for Natural Languages.* Kluwer, Boston, Mass.

Tanaka Hozumi: He is a professor of Department of Computer Science, Tokyo Institute of Technology. He received the B.S. degree in 1964 and the M.S. degree in 1966 from Tokyo Institute of Technology. In 1966 he joined in the Electro Technical Laboratories, Tsukuba. He received the Dr. Eng. degree in 1980. He joined in Tokyo Institute of Technology in 1983. He has been engaged in artificial intelligence and natural language processing research.

Tokunaga Takenobu: He is an associate professor of Department of Computer Science, Tokyo Institute of Technology. He received the B.S. degree in 1983 from Tokyo Institute of Technology, the M.S. and the Dr. Eng. degrees from Tokyo Institute of Technology in 1985 and 1991, respectively. His current interests are natural language processing, information retrieval.

Aizawa Michio: He is a researcher at Media Technology Laboratory, Canon Inc. He received the B.S. degree in 1991 from Tokyo Institute of Technology, the M.S. degrees from Tokyo Institute of Technology in 1993. His current interest is natural language processing.

(Received September 26, 1994)
(Revised December 22, 1994)
(Accepted January 13, 1995)