人工知能学会誌 Vol. 11 No. 2 (March 1996) 〔別刷〕

論 文

Technical Papers

Incorporation of Phoneme-Context-Dependence into LR Table through Constraint Propagation Method

Hozumi Tanaka* Hui Li* Takenobu Tokunaga*

* Dept. of Computer Science, Tokyo Institute of Technology, Tokyo 152, Japan.

Received December 14, 1994.

Keywords: allophone model, GLR parser, constraints propagation, speech recognition.

Summary

It is obvious that successful speech recognition requires the use of linguistic information. For this purpose, a generalized LR (GLR) parser provides an exceptionally competent and flexible framework to combine linguistic information with phonological information.

The combination of a GLR parser and allophone models is considered very effective for enhancing the recognition accuracy in a large vocabulary continuous speech recognition. The main problem of integrating GLR parsing into an allophone-based recognition system is how to solve the word juncture problem, that is, how to express the phones at a word boundary with allophone models.

This paper proposes a new method called CPM (Constraint Propagation Method) to generate an allophone-based LR table, which can effectively solve the word juncture problem. In our method, by introducing the allophone rules into the syntactic and lexical rules, an LR table is generated, then this LR table is modified on the basis of an allophone connection matrix by applying the constraint propagation method. With this modified LR table, precise allophone predictions for speech recognition can be obtained.

1. Introduction

It is obvious that successful speech recognition requires the use of linguistic information. For this purpose, a generalized LR (GLR) parser provides an exceptionally competent and flexible framework to combine linguistic information with phonological information.

One difficulty with large vocabulary continuous speech recognition is to reduce the search space. The GLR parser can meet this requirement [Kita 89] by applying linguistic constraints to speech recognition. In the phone-based speech recognition system, the GLR parser has been employed as a phoneme predictor, which provides efficient search of phones during the process of speech recognition. The GLR parser [Tomita 86] is guided by an LR table automatically generated from context-free grammar (CFG) rules and proceeds left-to-right without backtracking. In order to make phone predictions, the lookahead symbols in LR table are phones instead of the usual grammatical categories. Thus, lexical rules are expressed as follows:

⟨*grammatical category*⟩
→⟨*a sequence of phones*⟩.

On the other hand, several experiments [Itou 92, Lee 89, Nagai 93, Schwartz 85] have shown that the performance of speech recognition systems could be improved by using allophones as recognition units instead of phones. Allophone models (such as triphone models) are context-dependent phone models that take into consideration both the left and right neighboring phones to model the

major coarticulatory effects in continuous speech.

The combination of allophone models and a GLR parser [Itou 92, Nagai 93] is desirable to achieve better performance in continuous speech recognition. The main problem of integrating GLR parsing into an allophone-based recognition system is how to solve the word juncture problem, that is, how to express the phones at a word boundary with allophone models.

In this paper, we propose a new method to generate an allophone-based LR table that can solve the word juncture problem. This method, by introducing a set of allophone rules into the syntactic and lexical rules, generates an LR table, then modifies this LR table on the basis of an allophone connection matrix by applying the constraint propagation method.

The organization of this paper is as follows. **Chapter 2** provides an overview of the past allophone-based GLR parsing methods and points out problems in those methods; After discussing the advantages of using the canonical LR table for speech recognition, **Chapter 3** describes our method to generate an allophone-based LR table by applying CPM. **Chapter 4** provides comparisons between the LR tables before and after CPM; and **Chapter 5** concludes with the future works.

In the following chapters, we will use several examples from Japanese, but the method we propose is not language specific—it can be applied to many spoken languages.

2. Overview of Allophone-Based GLR Parsing Method

The main problem of integrating GLR parsing into an allophone-based recognition system is solving the word juncture problem. Consider a Japanese word "a k i (autumn)", which is a sequence of three phones in the lexical rule that follows:

$$noun \rightarrow a\ k\ i$$

The allophone corresponding to the phone "k", can be determined by the left and right context, which are known in advance, namely "a" and "i", respectively. On the other hand, the phones "a" and "i" are located at the word boundary. For the beginning phone "a", there is no left context, and for the end phone "i", there is no right context, and thus, for the phones at word boundaries, it is difficult to know their left or right contexts beforehand. To solve this problem, several allophone-based GLR parsing methods have been proposed [Itou 92, Nagai 93].

Itou, *et al.* [Itou 92] have used the lexical rules that follow:

$$noun \rightarrow a(*, k)\ k1\ i(k, *)$$

where "k1" is an allophone of "k", which has the left and right context "a" and "i"; "a(*, k)" is a special phone of "a" whose right context is known, and "i(k, *)" is a special phone of "i" whose left context is known. The LR table is constructed from a syntactic and lexical rule set. The allophones at word boundaries are determined dynamically during the recognition process when the preceding or succeeding words are obtained.

In this method, some changes in a GLR parsing algorithm are required to take account of the phoneme-context-dependence at word boundaries. Furthermore, for the end phones "i(k, *)" in the above example, the system makes needless allophone predictions because of no right context.

Nagai, *et al.* [Nagai 93] have proposed the three approaches that follow:

(*1*) *Grammar level realization* As well as Itou's method, phones within a word are changed into allophones. Phones at a word boundary are changed into possible allophones, which generate new grammatical categories. For instance, if "a(*, k)" and "i(k, *)" both have two allophones, "a1" and "a2", "i1" and "i2" respectively, then the four lexical rules are created from a word such as "a k i" are as follows:

$$a1\_noun\_i1 \rightarrow a1\ k1\ i1$$
$$a2\_noun\_i1 \rightarrow a2\ k1\ i1$$
$$a1\_noun\_i2 \rightarrow a1\ k1\ i2$$
$$a2\_noun\_i2 \rightarrow a2\ k1\ i2$$

Therefore, too many lexical rules are created from each word. The creation of new grammatical categories will produce many new syntactic rules. Furthermore, nonterminal symbols of RHS (right hand side) in the syntactic rule must take account of the word juncture problem considering newly created nonterminal symbols. For example, the nonter-

minal symbols, "i_cat_j" and "j_cat'_k" can be adjacent in this order, and so on. Thus, phoneme-context-dependence is expressed by a large number of lexical and grammar rules. For instance, in the case of 1 353 phoneme-context-independent CFG rules and 300 allophone models, the number of CFG rules becomes 61 507 [Nagai 93]*1. Although this method requires no change of a GLR parsing algorithm, the explosion of states in an LR table may occur.

(2) *Table level realization* From a set of syntactic and phoneme-context-independent lexical rules, this method generates an LR table, then it introduces a set of allophones step by step by adding new states and new nonterminal symbols in the LR table to incorporate phoneme-context-dependence. The table modification process is complex and requires changes to the parsing algorithm to keep the left and right context of allophones. Thus, phoneme-context-dependence must be dynamically recognized in the parsing process. This method results in the increase of the number of states in the LR table and brings inefficiency to the parsing process.

(3) *Parsing level realization* In this method, the phoneme-context-dependence is not incorporated in an LR table. The determination of allophones is completely handled by a GLR parser, which has to include a method of the phoneme-context-dependence in a procedural way. This makes the original GLR parsing algorithm more complex and inefficient.

The CPM proposed in this paper, by introducing a set of allophone rules and applying the constraints propagation, compiles the phoneme-context-dependence into an LR table in advance. In the CPM, the dynamic processing during the parsing, like that in the table and the parser level realizations [Nagai 93], is not required, therefore, no change in the GLR parsing algorithm is required. In contrast to the grammar level realization [Nagai 93], as CFG rules we use in our method, is equal to the number of phoneme-context-independent CFG rules plus the number of the allophone models, the explosion of CFG rules does not occur.

*1 According to our method explained later, the number of CFG rules is only 1 353+300(=1 653).

3. Algorithm for Generating Allophone-Based LR Table through Constraint Propagation

In this chapter, we propose a new method called CPM (Constraint Propagation Method) to generate a phoneme-context-dependent (allophone-based) LR table that can solve the word juncture problem and enable us to make precise allophone predictions.

The outline of our method is shown in Fig. 1.

In this method, from the given allophone models, an allophone connection matrix and a set of allophone rules are constructed, and for the lexical rules, the phones within a word are changed into the allophones. From a set of syntactic, lexical and allophonic rules (CFG), an initial allophone-based LR table is generated, and then this LR table is modified by using the allophone connection matrix through constraints propagation method.

Before explaining the details of each step, we would like to discuss the types of LR tables. All methods mentioned in **Chapter 2** have used the SLR or LALR table. Compared with the canonical LR table, the SLR and LALR table can not provide the precise phoneme predictions because the SLR and LALR table have fewer states due to merging several states in an LR table [Aho 86], and merging several states brings many actions in a state that produces many predictions.

This is why our method uses the canonical LR table. Generally, the canonical LR table has more states than the SLR or LALR table, but by applying CPM, we achieve reduction of the table size.

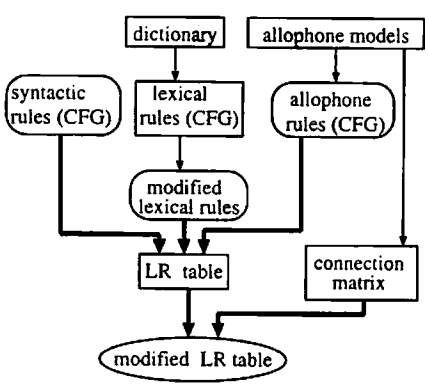

Fig. 1 Outline of generating an allophone-based LR table.

(1) S → N BE (3) N → ch i ch i

(2) N → h a h a (4) BE → d a

Fig. 2 An example of syntactic and lexical rule set (CFG).

In this chapter, we will use an example of simple Japanese syntactic and lexical rules (CFG) shown in Fig. 2 to illustrate our method.

3・1 *Allophone connection matrix*

The allophone context of an allophone "x" is defined as:

⟨left context⟩x⟨right context⟩

where *⟨left context⟩* and *⟨right context⟩* are a set of phones. We assume that there is only one allophone context for one allophone. An allophone connection matrix is created from a set of allophone contexts.

Let us consider allophone "i2" of phone "i", allophone "d1" of phone "d", and the allophone contexts shown below.

$$\left\{\begin{matrix}\vdots\\ ch\\ \vdots\end{matrix}\right\} i2 \left\{\begin{matrix}\vdots\\ d\\ \vdots\end{matrix}\right\}, \quad \left\{\begin{matrix}\vdots\\ i\\ \vdots\end{matrix}\right\} d1 \left\{\begin{matrix}a\\ \vdots\\ \vdots\end{matrix}\right\}$$

According to the above allophone contexts in the form of triphone, "d1" can follow "i2" because the right context of "i2" contains the phone "d" and the left context of "d1"contains the phone "i".

If a connection matrix is expressed as an array of *Connect* [left_allophone, right_allophone], we can fill *Connect* [i2, d1] with symbol "1" to indicate that "i2" and "d1" are connectable in this order. Therefore, we can construct an allophone connection matrix from a set of allophone contexts. Note: an entry in the matrix, which can not be filled with "1", is marked by "0" to indicate the nonconnectability between the corresponding two adjacent allophones.

Fig. 3 is an example of the allophone connection matrix partially filled. We will use this connection matrix to incorporate allophone connection constraints into the LR table at Sec. 3・4 by applying CPM.

3・2 *Modification of the lexical rules*

The phones within a word are automatically converted into the allophones using a set of allophone contexts. We can not, however, change the

		RIGHT										
		h1	h2	a1	a2	ch1	ch2	i1	i2	d1	d2	d3	$
LEFT	h1			1	0								
	h2			0	0								
	a1	1	1							1	0	1	0
	a2	0	0							0	0	0	1
	ch1							0	0				
	ch2							0	1				
	i1					0	0			0	0	0	
	i2					1	1			0	1	1	
	d1			0	1								
	d2			0	1								
	d3			0	0								

Fig. 3 An example of allophone connection matrix partially filled.

(2)' N → h a1 h1 a (4)' BE → d a

(3)' N → ch i2 ch2 i

Fig. 4 Conversion of the lexical rules.

phones at word boundaries because of the word juncture problem. Therefore, the lexical rules (rule (2) to (4) in Fig. 2) are changed into those shown in Fig. 4.

3・3 *Allophone-based LR table*

The allophone rules are derived by pairing a phone with the corresponding allophones:

⟨phone⟩→⟨allophone1⟩|⟨allophone2⟩|⋯

Assume the following: {h1, h2} for "h", {a1, a2} for "a", {ch1, ch2} for "ch", {i1, i2} for "i", {d1, d2, d3} for "d", a set of allophone rules from (5) to (14) in Fig.

(5) h → h1 (9) ch → ch1 (13) d → d1

(6) h → h2 (10) ch → ch2 (14) d → d2

(7) a → a1 (11) i → i1 (15) d → d3

(8) a → a2 (12) i → i2

Fig. 5 A set of allophone rules.

(1) S → N BE (9) ch → ch1

(2)' N → h a1 h1 a (10) ch → ch2

(3)' N → ch i2 ch2 i (11) i → i1

(4)' BE → d a (12) i → i2

(5) h → h1 (13) d → d1

(6) h → h2 (14) d → d2

(7) a → a1 (15) d → d3

(8) a → a2

Fig. 6 A set of syntactic, lexical and allophonic rules.

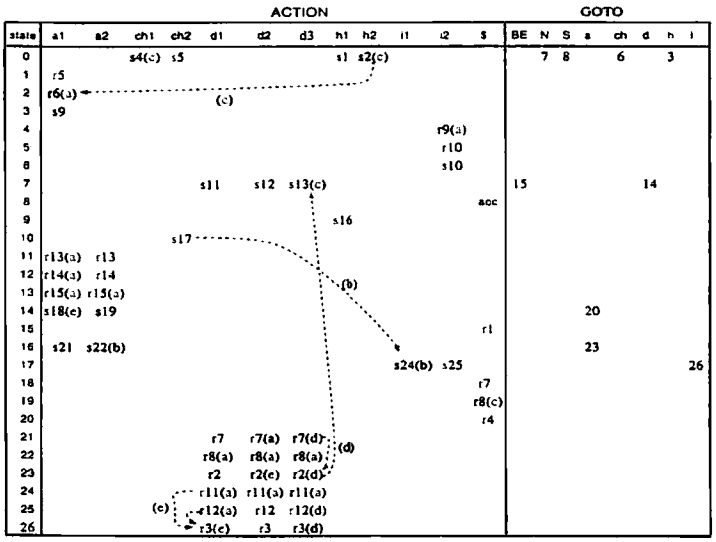

Fig. 7 Canonical LR (CLR) table generated from rules in Fig. 6.

```
for each reduce action R with an allophone rule in each entry of LR table {
    if (Connect[x, y] = 0) {
        mark R "deletable";
    }
}
where
 x: the RHS of the allophone rule used by R.
 y: the lookahead symbol of R.
 Connect: the allophone connection matrix.
```

Fig. 8 Remove the illegal reduce actions with allophone rules.

5 can be produced.

Now we have a set of syntactic, lexical and allophonic rules as shown in Fig. 6.

From the above extended CFG rules, we can generate a canonical LR table as shown in Fig. 7. Note that all the lookahead symbols in this table are allophones.

In the extended CFG rules in Fig. 6, the information about connectability represented in Fig. 3 is not included for the phones at the word boundaries. For example, rules (2)′, (6) and (8) allow the sequence of allophones "h2 a1 h1 a2" for the word "h a h a (mother)" which violates the connectability between "h2" and "a1", "h1" and "a2" as shown in Fig. 3.

3・4 Modifications of LR table

In order to incorporate allophone connection constraints into the LR table, we combine our CPM with the method developed by Tanaka, *et al.* [Tanaka 93] and modify the original canonical LR table. Initial constraints are imposed on the LR table by using allophone connection matrix, then these constraints propagate throughout the LR table to produce a modified allophone-based LR table.

〔1〕 Connection check

At first, the constraints on connectability are introduced into the LR table by deleting illegal actions which violate the connection constraints represented in connection matrix.

(a) Deletable reduce action with an allophone rule In this step, we check every reduce action with an allophone rule by employing the connection matrix in Fig. 3. In a similar way as [Tanaka 93], the illegal reduce actions with allophone rules, which violate connection constraints, are marked "deletable". Fig. 8 shows this procedure.

Consider, for example, re6 with lookahead symbol

```
for each shift action S in each entry of LR table {
  if (the action prior to S is a shift action) {
    if (Connect[x, y] = 0) {
      mark S "deletable";
    }
  }
}
where
 x: the lookahead symbol of the shift action prior to S.
 y: the lookahead symbol of S.
```

Fig. 9 Remove the illegal shift actions by connection matrix.

"a1" at state 2 in Fig. 7. The allophone connection matrix indicates that the connection between "h2" (RHS of rule 6) and "a1" is not allowed (*Connect* [h2, a1]=0), so this reduce action is illegal, and is marked "deletable".

In Fig. 7, all the deletable reduce actions found by this step are marked (a).

(b) Deletable shift action whose predecessors are shift actions Let us consider the left of the end phone of a word. If the left phone is an allophone, we can easily check the connectability between the left allophone and its succeeding end phone. In this case, after shifting the left allophone, we have to immediately shift all the possible allophones belonging to this end phone. Consecutive shift actions will occur.

Consider a Japanese word "ch i2 ch2 i", and assume that there are two possible allophones "i1" and "i2" for the end phone "i". The left allophone of the end phone "i" is "ch2". After shifting "ch2" by sh17 in state 10, we have to shift "i1" and "i2". In Fig. 7, sh24 in state 17 is to shift "i1". *Connect* [ch2, i1]=0 means, however, that shifting "i1" is not allowed. Therefore, sh24 in state 17 is not allowed and is marked "deletable". On the contrary, sh25 with lookahead symbol "i2" in state 17 is not "deletable", because *Connect*[ch2, i2]=1. Fig. 9 shows this procedure.

In Fig. 7, all the deletable shift actions found by this step are marked (b).

[2] Constraints propagation

Secondly, the above constraints propagate throughout the LR table by deleting all the other illegal actions.

(c) Deletable shift action that lead to an empty state An empty state is defined as a state whose actions are all marked "deletable". For an empty state, the shift actions that lead to this empty state should be marked "deletable". Meanwhile, for a state, if all of its preceding actions are marked "deletable", all the actions in this state should be marked "deletable" too.

For example, in Fig. 7, sh2 in state 0 with the lookahead symbol "h2" should be marked "deletable", since state 2 is an empty state (the unique action re6 in state 2 has been marked "deletable" by step (a)).

In Fig. 7, all the deletable actions found by this step are marked (c).

(d) Deletable reduce action whose successors are all "deletable" actions For a reduce action, after it has been carried out, the parser will transfer to the new states according to the goto graph, and the new actions with the same lookahead symbol in the new states will be carried out next. If all the new

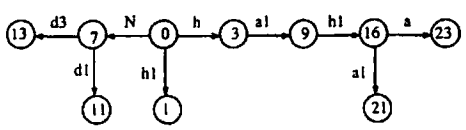

Fig. 11 A part of the goto graph reconstructed from Fig. 7.

```
for each reduce action R in each entry of LR table {
  if (every action that R will lead to has already been marked "deletable") {
      mark R "deletable";
  }
}
```

Fig. 10 Remove the reduce actions whose succeeding actions have been all marked "deletable".

```
for each action A in each entry of LR table {
  if (all the preceding actions of A have been marked "deletable") {
      mark A "deletable";
  }
}
```

Fig. 12 Remove the actions whose preceding actions have been all marked "deletable".

actions in the new states have been marked "deletable", the reduce action R should be marked "deletable" too.

We provide this procedure in Fig. 10.

Fig. 11 illustrates a part of the goto graph reconstructed from the LR table in Fig. 7.

Consider re7 in state 21 with a lookahead symbol "d3" in Fig. 7. According to the above goto graph, the parser will transfer to state 23 after re7, applying rule 7 (a → a1) in state 21*2 Thus, in state 23, the next reduce action becomes re2 (N → h a1 h1 a) with the same lookahead symbol "d3". Therefore, re7 (lookahead symbol "d3") in state 21 will lead to re2 (lookahead symbol "d3") in state 23.

After re2, the parser will transfer to state 7, since from state 23 we can traverse the goto graph in reverse such as "a", "h1", "a1", "h" and reaches state 0 from where it transfers to state 7 by shifting N. In state 7, the action with the same lookahead symbol "d3" is sh13, so re2 (lookahead symbol "d3") in state 23 will lead to sh13 in state 7.

However, in state 7, sh13 has already been marked "deletable" by the step (c). Thus, re2 (lookahead symbol "d3") in state 23 should be marked "deletable", then re7 (lookahead symbol "d3") in state 21 should be marked "deletable" too.

In Fig. 7, all the deletable reduce actions found by this step are marked (d).

(e) Deletable action whose predecessors are all "deletable" actions For an action, if all the actions that will lead to this action have been marked "deletable", this action should be marked "deletable" too. Fig. 12 shows this procedure.

For example, the reduce action re3 with the lookahead symbol "d1" in state 26 is transferred from re11 in state 24 and re12 in state 25 with lookahead symbol "d1", but these two reduce actions (re11 and re12) have been marked "deletable" by step (a), so this reduce action re3 (lookahead symbol "d1") should also be marked "deletable".

This constraints propagation (step (c)～(e)) should be repeated until no more "deletable" actions are found, and then the LR table is compressed by deleting all the "deletable" actions and all the empty states to reduce the table size.

We summarized our algorithm in Fig. 13.

```
CPM()
{
  /* connection check */
    step (a) and (b);
  /* constraints propagation */
  while(1) {
    step (c) to (e);
    if(no new "deletable" action comes out)
      break;
  }
  compact LR table;
}
```

Fig. 13 A top level procedure of CPM.

	ACTION								GOTO					
state	a1	a2	ch2	d1	d2	h1	i2	$	BE	N	S	a	ch	d	h	i
0			s5			s1				7	8		6		3	
1	r5															
3	s9															
5							r10									
6							s10									
7				s11	s12				15					14		
8								acc								
9						s16										
10			s17													
11	r13															
12	r14															
14	s19												20			
15								r1								
16	s21												23			
17							s25									26
18								r7								
20								r4								
21				r7												
23				r2												
25					r12											
26					r3											

Fig. 14 The modified canonical LR (MCLR) table.

* 2 The lookahead symbol, "d3", remains the same during the reduce action.

The above procedure enables us to introduce the phoneme-context-dependence into the phones at word boundaries and solve the word juncture problem.

Fig. 14 is the modified allophone-based canonical LR (MCLR) table from Fig. 7 after applying CPM.

4. The Effects of CPM

Compared with the table of Fig. 7, the lookahead symbols in Fig. 14 includes only "a1", "a2", "ch2", "d1", "d2", "h1" and "i2". The allophones at word boundaries are restricted by CPM through deleting the illegal actions. Generally, the allophones at a word boundary are not uniquely determined before the preceding or succeeding phones are known. This very simple example suggests how the word juncture problem can be solved by applying CPM.

For a task with 64 syntactic rules and 120 lexical rules, the canonical LR (CLR) tables before and after applying CPM are compared with the number of states and actions. The syntactic rules, lexical rules and allophone models we used are the same as in [Itou 92], which have already been used in a speech recognition system.

Table 1 shows the table size before and after CPM for 128, 256, 512 and 1 024 allophone models. After applying CPM, in the case of 128 allophone models, the number of states, shift actions, and reduce actions decreases to 61.4%, 38.6%, and 11% of the original canonical LR table, and in the case of 1 024 allophone models, the number of states, shift actions, and reduce actions decreases to 11%, 4.9%, and 0.2% of the original canonical LR table. The more the number of allophones are, the more the table size decreases.

Table 1 Comparison of table size before and after CPM.

allophones	table	state	shift	reduce	goto
128	CLR	1722	2260	9254	421
	MCLR	1016	873	1015	421
256	CLR	2959	4760	37831	421
	MCLR	1061	930	1072	421
512	CLR	5627	10211	161127	421
	MCLR	1096	971	1135	421
1024	CLR	11157	21057	701370	421
	MCLR	1139	1003	1212	421

Reduction of the reduce and shift actions implies that the allophone predictions in speech recognition become more accurate.

5. Discussions and Conclusions

We have proposed a new method to generate an allophone-based LR table that solves the word juncture problem and enables to predict the allophones precisely in speech recognition. In this method, by introducing the allophone rules into the syntactic and lexical rules, an initial LR table is generated, then, on the basis of an allophone connection matrix, the LR table is modified by applying CPM.

The characteristics of our approach can be summarized as follows :

(1) Generation of allophone-based LR table is realized at two levels, this enables us to use the existing LR table generation algorithms to generate an initial allophone-based LR table.

(2) By introducing an allophone connection matrix and applying CPM, a large number of illegal states and actions of initial LR table can be deleted, which results in enormous reduction of the table size, and at the same time the word juncture problem can be solved. Furthermore, the reduction of actions and states provides us with accurate allophone predictions in speech recognition.

(3) In our method, solving the word juncture problem does not need change in the GLR parsing algorithm, since the connection constraint between two adjacent allophones has been incorporated into an LR table in advance.

Although the proposed method has the advantage of enabling us to use already existing LR table generation method to get the initial LR table, the number of states and actions of initial LR table often explodes as the number of syntactic and lexical rules or allophone models increases. In order to rectify this situation, we can modify the LR table generation algorithm slightly in order to incorporate the allophone connection constraints (step (a)

and (b) in Chapter 3) during the generation process. In the case of 1 024 allophones for the above example grammar, the number of the states of the original LR table decreases to 1/6 by changing the LR table generation algorithm.

The future works will include incorporating the LR table generated by CPM into an allophone-based continuous speech recognition system.

Acknowledgment

The authors are grateful to Dr. S. Hayamizu and Dr. K. Itou for their valuable discussions and supports. They are also grateful to Mr. H. Nakajima for supplying us the program of generating canonical LR table.

◇ References ◇

[Aho 86] Aho, A., Ravi, S. and Ullman, J.: *Compilers: Principle, Techniques, and Tools*, Addision-Wesley (1986).

[Itou 92] Itou, K., Hayamizu, S. and Tanaka, H.: Continuous Speech Recognition by Context-Dependent Phonetic HMM and an Efficient Algorithm for Finding N-Best Sentence Hypotheses, *Proc. ICASSP-92*, pp. I-21-I-24 (1992).

[Kita 89] Kita, K., Kawabata, T. and Saito, H.: HMM Continuous Speech Recognition Using Predictive LR Parsing, *Proc. ICASSP-89*, pp. 703-706, IEEE (1989).

[Lee 89] Lee, K. F.: *Automatic Speech Recognition: The Development of the SPHINX System*, Kluwer Academic Publishers (1989).

[Nagai 93] Nagai, A., Sagayama, S., Kita, K. and Kikuchi, H.: Three Different LR Parsing Algorithms for Phoneme-Context-Dependent HMM Based Continuous Speech Recognition, *IEICE Trans. Inf. & Syst.*, Vol. E76-D, No. 1, pp. 29-37 (1993).

[Schwartz 85] Schwartz, R., Chow, Y., Kimball, O., Roucos, S., Krasner, M. and Makhoul, J.: Context dependent modeling for acoustic phonetic recognition of continuous speech, *Proc. ICASSP-85*, pp. 1205-1208, IEEE (1985).

[Tanaka 93] Tanaka, H., Tokunaga, T. and Aizawa, M.: Integration of Morphological and Syntactic Analysis Based on LR Parsing Algorithm, *Proc. Int. Workshop on Parsing Technologies*, pp. 101-109, Tilburg (1993).

[Tanaka 94] Tanaka, H., Li, H. and Tokunaga, T.: Incorporation of Phoneme-Context-Dependence into LR Table through Constraint Propagation Method, *Proc. AAAI-94 Workshop on Integration of Natural Language and Speech Processing*, pp. 15-22, Seattle, Washington (1994).

[Tomita 86] Tomita, M.: *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems*, Kluwer Academic Publishers, Boston, MA (1986).

〔担当編集委員・査読者：中村貞吾〕

Authors' Profile

Hozumi Tanaka (Member)

He received the B.S. and M.S. degrees in faculty of science and engineering from Tokyo Institute of Technology in 1964 and 1966, respectively. In 1966 he joined in the Electro Technical Laboratories, Tsukuba. He received his Doctor of Engineering in 1980. In 1983 he joined as an associate professor in the faculty of Information Engineering in Tokyo Institute of Technology and he became professor in 1986. He has been engaged in Artificial Intelligence and Natural Language Processing research. He is member of the Information Processing Society of Japan, etc.

Hui Li (Student Member)

He received the B.E. and M.E. degrees in electrical engineering from University of Science and Technology of China in 1983 and 1986, respectively. He is currently a Ph.D. student in the Dept. of Computer Science at Tokyo Institute of Technology. His research interests include speech recognition and natural language processing. He is member of the Acoustic Society of Japan and the Information Processing Society of Japan.

Takenobu Tokunaga (Member)

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. He is an Associate Professor of Department of Computer Science, Tokyo Institute of Technology. His current interests are natural language processing, information retrieval.