3—8—9 An Efficient Phoneme-Context-Dependent LR Table and its Applications in Continuous Speech Recognition*

◎ Hui Li[1], Toshiyuki Takezawa[2], Harald Singer[2], △Teruaki Hayashi[2], △Hozumi Tanaka[1]
[1]Tokyo Institute of Technology [2]ATR Interpreting Telecommunications Research Labs.

1 Introduction

Integrating a GLR parser into an allophone-based recognition process is desirable to achieve better performance in continuous speech recognition[1]. In this case, GLR parser, driven by an LR table generated from a context-free grammar, is used as an allophone predictor. A major problem in integrating a GLR parser into an allophone-based recognition system is how to solve the word juncture problem.

In this paper, we describe a new method of generating an allophone-based LR table from a set of CFG, lexical and allophone rules by using constraint propagation method(CPM), and present some experimental results on continuous speech recognition.

2 Algorithm

The algorithm consists of four stages:

(1) Construction of allophone connection matrix
(2) Conversion of lexical rules
(3) Generation of allophone-based LR table
(4) Modifications of allophone-based LR table

1) Allophone connection matrix

The allophone connection matrix provides the connectability between two adjacent allophones.

For example, assume that the context sets of allophones "i2" for phone "i" and "d1" for phone "d" are as follows.

$$\text{i2}:\left\{\begin{array}{c}\vdots\\ ch\\ \vdots\end{array}\right\}\text{i}\left\{\begin{array}{c}\vdots\\ d\\ \vdots\end{array}\right\},\qquad \text{d1}:\left\{\begin{array}{c}\vdots\\ i\\ \vdots\end{array}\right\}\text{d}\left\{\begin{array}{c}a\\ \vdots\\ \vdots\end{array}\right\}$$

We say "d1" can follow "i2" because the right context of "i2" contains the phone "d" and the left context of "d1" contains the phone "i". A connection matrix is expressed as an array of *Connect*[left_allophone, right_allophone], whose value is "1" (two allophones are connectable) or "0" (two allophones are not connectable).

2) Conversion of the lexical rules

We use a simple Japanese CFG and lexical rules shown below to illustrate the conversion of the lexical rules and CPM algorithm of generating allophone-based LR table.

(1) S → N BE (3) N → ch i ch i
(2) N → h a h a (4) BE → d a

We change the phones within a word into the allophones according to the allophone contexts. Therefore, the lexical rules become those shown below.

(2)' N → h a1 h1 a (4)' BE → d a
(3)' N → ch i2 ch2 i

3) Allophone-based LR table

To solve the word juncture problem, we introduce the allophone rules into the set of CFG and lexical rules. The allophone rule is derived by pairing a phone with the corresponding allophones.

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 can be produced.

(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

From the set of CFG rule((1)), converted lexical rules((2)'-(4)') and allophone rules((5)-(15)), an allophone-based canonical LR table is generated. Fig. 1 shows part of this LR table.

state	action					
	h2	a1	ch2	i1	d2	d3
0	sh6(c)					
6*		re6(a)				
7					sh12(c)	sh13(e)
8			sh21			
18					re2(d)	re2(e)
19					re7(d)	re7(a)
21				sh23(b)		

Fig. 1 Part of an allophone-based LR table

4) Modifications of LR table

The LR table in Fig. 1 does not include any phoneme context information for the phones at word boundaries. In order to incorporate the connection constraints into the LR table, we modify the LR table by the following steps(we use Fig. 1 to explain the modifications of LR table later).

(1). Connection check

At first, we use connection matrix to remove the illegal actions of LR table in a similar way as in [2].

(a). Remove the illegal reduce actions of allophone rules

For example, re6(lookahead "a1") in state 6 should be removed if "h2" (RHS of rule 6) and "a1" are not connectable(*Connect*[h2,a1] = 0).

*"効率的な音素環境依存 LR テーブル作成法と連続音声認識システムへの応用", 李輝 (東工大), 竹沢寿幸 (ATR), シンガーハラルド (ATR), 林輝昭 (ATR), 田中穂積 (東工大)

(b). Remove the illegal shift actions whose predecessors are shift actions

For example, sh23(lookahead "i1") in state 21 should be removed if "ch2"(lookahead of preceding shift action) and "i1" are not connectable.

(2). Constraint propagation

Secondly, we remove all other illegal actions through constraint propagation.

(c). Remove the shift actions that lead to empty states

An empty state is defined as a state whose all actions have already been removed. The shift actions that lead to empty state should be removed. For example, state 6 is an empty state, so sh6(lookahead "h2") in state 0 should be removed.

(d). Remove the reduce actions that lead to removed shift actions

Consider re7(lookahead "d2") in state 19, the parser will transfer to state 18 after re7. In state 18, the next action is re2(lookahead "d2"), after re2, the parser will transfer to state 7, but in this state, sh12(lookahead "d2") has already been removed by the substep (c). Thus, re7 (in state 19) and re2 (in state 18) should be removed too.

(e). Remove the illegal actions whose predecessors are goto actions

Consider re2 (lookahead "d3") in state 18, this action is transferred from a goto action after re7 (lookahead "d3") in state 19. Since re7 (lookahead "d3") has been removed by substep(a), so re2 (lookahead "d3") in state 18 should be removed. It is the same to sh13(lookahead "d3") in state 7.

The algorithm will repeat this constraint propagation(substep(c) - (e)) until no more actions are removed, and then compact the LR table to reduce the table size.

3 Table Size

An ATR phrase grammar for the "conference registration task" with 1225 CFG rules and 1588 lexical rules is used. The phoneme perplexity of test set is 3.43. Allophone models are generated by the SSS algorithm[1]. Table 1 lists the size of LR table generated by our method. For the sake of comparison, the LR table size in the case of 26 phones is also listed. Comparing with the phoneme-context-independent(26 phones) LR table, in the case of 1759 allophones, the number of states of allophone-based LR table only increased by about 35%.

4 Recognition Experiments

We have carried out speech recognition experiments based on ATR's SSS-LR recognition system[1]. In ATR's SSS-LR system, phoneme-context-dependent parser is realized at the parser level. ATR's phrase grammar(described in section 3) is used. Table 2 lists the recognition rates of two kinds of GLR parser realizations: one is based on an allophone-based LR table generated by our method(CPM), the other is ATR's parser level realization. It can be seen that when beam width is 50, the recognition rates are slightly improved using our method. Fig. 2 shows a comparison of CPU time. Since the phoneme-context-dependency is compiled into the LR table in advance, less CPU time is required with our method.

5 Conclusions

We have described an algorithm for generating an allophone-based LR table through constraint propagation method, and experiment results have been provided. The future work will include:

(1) Sentence recognition

(2) Integration of morphological constraints into a LR table in addition to allophonic constraints

(3) Application to stochastic CFG.

References

[1] Nagai,A. et.al. *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, January, 1993

[2] Tanaka,H. et.al. *Integration of Morphological and Syntactic Analysis Based on LR Parsing Algorithm.* International Workshop on Parsing Technologies, Tilburg, pp.101-109, 1993

Table 1: Table size

allophones	26*	283	1026	1759
state	7411	9004	9628	9994
shift	6578	10696	14020	16035
reduce	15428	21145	23891	24602
goto	1463	4013	4013	4013

Table 2: Recognition rates

allophone number	method	beam=50		beam = 250	
		rank 1	rank 5	rank 1	rank 5
26*		82.90	90.43	87.83	97.10
283	ATR	83.48	90.72	87.83	97.10
	CPM	84.06	91.30	87.83	97.39
1026	ATR	88.12	95.07	91.01	99.13
	CPM	88.70	95.94	91.01	99.13

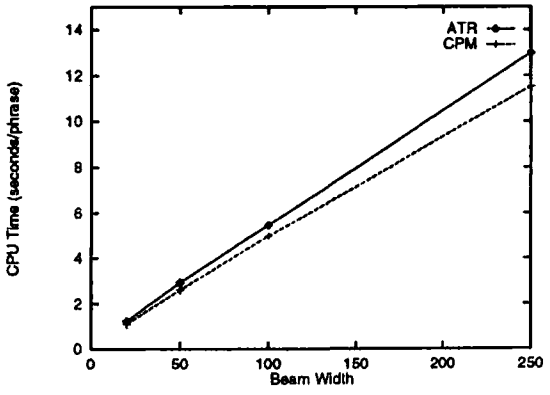

Fig. 2 Comparison of CPU time