TR-IT-0058

# An Efficient Phoneme-Context-Dependent LR Table Applied to the SSS-LR Continuous Speech Recognition System

李 輝
Hui Li

竹沢 寿幸
Toshiyuki Takezawa

シンガー ハラルド
Harald Singer

林 輝昭
Teruaki Hayashi

1994.06

In this report, we describe a method of generating a phoneme-context-dependent canonical LR table through constraint propagation method (CPM) and its application to the SSS-LR continuous speech recognition system. For phrase recognition tasks, a comparison of recognition performance for table level and parser level realization of GLR parsing algorithm is given. We also compare the performance of SLR and CLR table for speech recognition.

# Contents

# List of Figures



# List of Tables

# 1 Introduction

One difficulty with large vocabulary continuous speech recognition is to reduce the search space. The GLR parser can meet this requirement [4][3] 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 in the process of speech recognition. The GLR parser is guided by an LR table automatically generated from context-free grammar (CFG) rules and proceeds left-to-right without backtracking [15].

On the other hand, in continuous speech recognition, the performance of speech recognition systems has been improved by using allophones as recognition units instead of phones [3][6][8][11]. Allophone models (such as triphone models) are context-dependent phone models that take into consideration the left and right neighboring phones to model the major coarticulatory effects in continuous speech.

The combination of allophone models and a GLR parser is desirable to achieve better performance in continuous speech recognition. At ATR, a highly accurate speech recognition system, called SSS-LR, which integrates HMM-derived allophone models and a phoneme-context-dependent LR parser, has been realized.

One of the difficulties of integrating GLR parser with an allophone-based HMM recognition system is how to solve the word juncture problem, because for the phones at word boundaries, their left or right context depends on the preceding and succeeding words.

Generally, there are three possible approaches to realize the phoneme-context-dependent GLR parser, namely, grammar level realization, table level realization and parser level realization.

Parser level realization makes original GLR parsing algorithm more complex and inefficient, because phone context is decided dynamically during the recognition process.

Grammar level and LR table level realization is supposed to be more efficient than parser level realization, since the information of phonetic contexts are compiled in the LR parsing table in advance,

To construct phoneme-context-dependent LR table, an efficient method, called CPM (constraint propagation method), has been proposed by Tanaka et.al. [14].

In this report, we will describe the incorporation of the phoneme-context-dependent LR table generated by CPM into ATR SSS-LR continuous speech recognition system.

The report is organized as follows: Section 2 describes the SSS-LR continuous speech recognition system. Section 3 discusses the two different type of LR tables (SLR table and canonical LR table) applied to speech recognition. Section 4 describes the CPM method of generating allophone-based LR table; Section 5 presents the GLR parsing algorithm we used. Section 6 provides some experiment results. Section 7 concludes with discussions.

# 2 SSS-LR Continuous Speech Recognition System

The SSS-LR system [7][10], which integrates the allophone HMM model and phoneme-context-dependent LR parser, is a highly accurate continuous speech recognition system.

Fig. 1 shows the block diagram of SSS-LR system.

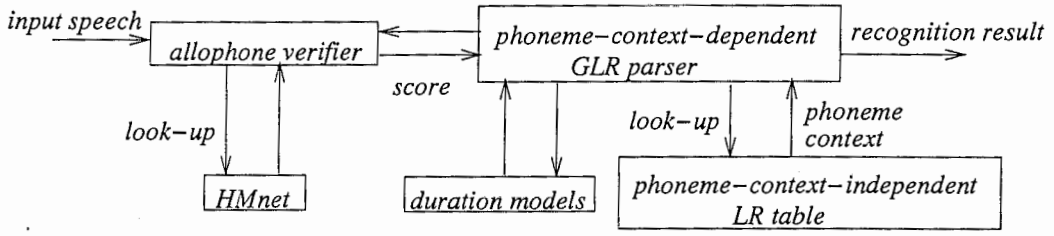



Figure 1: SSS-LR continuous speech recognition system

In this system, allophone models are represented efficiently as a shared-state network automatically generated by the Successive State Splitting (SSS) algorithm. The phoneme-context-dependent LR parser was used to drive allophone verifier.

The phoneme-context-dependent LR parser of SSS-LR is realized at the parser level, and phoneme context is dynamically decided during the recognition process by using a phoneme-context-independent LR table.

For example, consider the following LR table:

| | a | n | i | e |
|---|---|---|---|---|
| 0 | sh1 | | | |
| 1 | | sh2 | | |
| 2 | | | sh10 | sh11 |

If the input symbol is "n" in state 1, the parser acts as follows:

- gets the preceding phone "a" through looking back at state 0.
- gets the succeeding phone "i" and "e" through looking ahead at state 2.
- constructs phoneme triplets "a/n/i" and "a/n/e", then according to the allophone context set, renames these two triplets with allophones, for example "n1" for "a/n/i", "n2" for "a/n/e", and verifies them with allophone HMM model.

This method makes the original LR parser algorithm more complex, and requires additional computation during the recognition process.

# 3 SLR and Canonical LR Table

In this section, we would like to discuss the types of LR tables. All methods mentioned in Section 2 have used the SLR or LALR table. Compared with the canonical LR table, the SLR and LALR table can not provide us precise phoneme predictions because the SLR and LALR table have fewer states due to merging several states in an LR table [1], and merging several states brings many actions in a state that produces many predictions.

Consider the following simple grammar.

(1) <s> → <np>
(2) <np> → <n> <p>
(3) <np> → <adj> <n>
(4) <n> → h o N
(5) <p> → o
(6) <adj> → i i

Figure 2: A simple grammar

According to this grammar, there are only two correct phrases "h o N / o (本を)" and "i i / h o N (いい本)".

The SLR table and canonical LR table generated from this grammar are shown in Fig. 3 and Fig. 4 respectively.

| | ACTION | | | | | GOTO | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| state | h | o | N | i | $ | <n> | <adj> | <np> | <p> | <s> |
| 0 | sh1 | | | sh3 | | 2 | 4 | 5 | | 6 |
| 1 | | sh7 | | | | | | | | |
| 2 | | sh8 | | | | | | | 9 | |
| 3 | | | | sh10 | | | | | | |
| 4 | sh1 | | | | | 11 | | | | |
| 5 | | | | | re1 | | | | | |
| 6 | | | | | acc | | | | | |
| 7 | | | sh12 | | | | | | | |
| 8 | | | | | re5 | | | | | |
| 9 | | | | | re2 | | | | | |
| 10 | re6 | | | | | | | | | |
| 11 | | | | | re3 | | | | | |
| 12 | | re4 | | | re4 | | | | | |

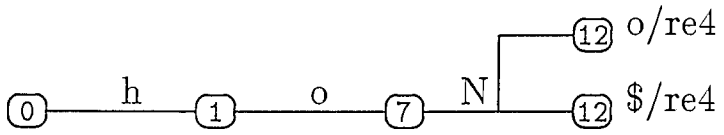
Figure 3: SLR table generated from the grammar in Fig. 2

In this example, the SLR table merged two states (state 13 and state 14 in Fig. 4) of the canonical LR table into one state (state 12 in Fig. 3). This merge will bring about useless phone predictions in speech recognition.

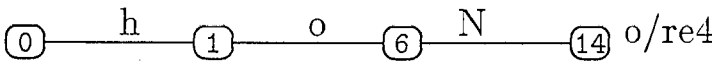
| | ACTION | | | | | GOTO | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| state | h | o | N | i | $ | <n> | <adj> | <np> | <p> | <s> |
| 0 | sh1 | | | sh2 | | 5 | 3 | 4 | | 15 |
| 1 | | sh6 | | | | | | | | |
| 2 | | | | sh7 | | | | | | |
| 3 | sh8 | | | | | 9 | | | | |
| 4 | | | | | re1 | | | | | |
| 5 | | sh11 | | | | | | | 10 | |
| 6 | | | sh14 | | | | | | | |
| 7 | re6 | | | | | | | | | |
| 8 | | sh12 | | | | | | | | |
| 9 | | | | | re3 | | | | | |
| 10 | | | | | re2 | | | | | |
| 11 | | | | | re5 | | | | | |
| 12 | | | sh13 | | | | | | | |
| 13 | | | | | re4 | | | | | |
| 14 | | re4 | | | | | | | | |
| 15 | | | | | acc | | | | | |

Figure 4: Canonical LR (CLR) table generated from the grammar in Fig. 2

For example, if the correct phrase is "h o N / o", using the SLR table, we have a path:




In state 12, an useless prediction "$" occured.
But if using the CLR table, the path is:




The useless prediction "$" did not occur.

In particular, for the phoneme-context-dependent LR parser, this useless prediction may affect the phone context of the preceding and succeeding phones. In the above example, for "N" of "h o N / o", using SLR table, we will get the two triplets "o/N/o" and "o/N/$", where "o/N/$" is an useless triplet, but using CLR table, "o/N/$" does not occur.

Generally, the canonical LR table has more states than the SLR or LALR table.

Table 1 shows a comparison of table size between SLR and CLR table for two phrase grammars, mset_phrase (979 rules) and eset_phrase (2813 rules), used at ATR.

Table 1: Comparison of table size between SLR and CLR table

| grammar | table type | state | shift | reduce | goto | predictions/state |
|---|---|---|---|---|---|---|
| mset_phrase | SLR | 2171 | 1825 | 3474 | 691 | 2.44 |
| mset_phrase | CLR | 2317 | 1920 | 3564 | 700 | 2.36 |
| eset_phrase | SLR | 6556 | 5967 | 14628 | 1409 | 3.14 |
| eset_phrase | CLR | 7411 | 6578 | 15428 | 1463 | 2.97 |

Compared with the SLR table, the number of states increased by 6.7% (mset_phrase) and 13% (eset_phrase), but the average phone predictions for each state decreased. This will result in the precise phone prediction for speech recognition. We will compare the application of these two types of LR table to the SSS-LR recognition system in Section 6.

# 4 CPM Method of Generating Allophone-Based LR Table

In this section, we describe the CPM method of generating allophone-based LR table.
This method consists of three steps:
(1) Construction of an allophone connection matrix that provides the connectability between two adjacent allophones.
(2) Conversion of the lexical rules and introduction of allophone rules.
(3) Modification of allophone-based LR table through constraint propagation.
The outline of CPM method is shown in Fig. 5.

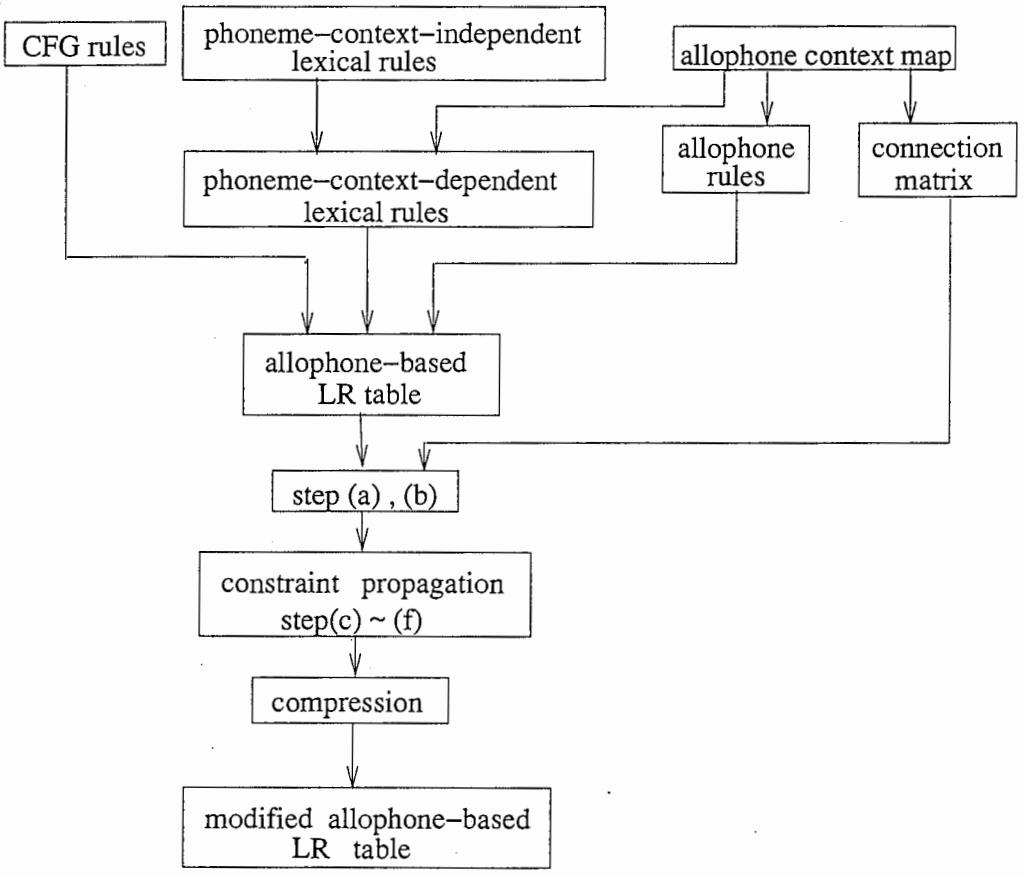



Figure 5: Outline of CPM

## 4.1 Construction of the Allophone Connection Matrix

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

$$<\textit{left context}>\ x\ <\textit{right context}>$$

where <*left context*> and <*right context*> are a set of phones. An allophone connection matrix is created from a set of allophone contexts.

Consider allophone "i2" of phone "i", allophone "d1" of phone "d", and the allophone contexts shown below.

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

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".

In this way, we can construct an allophone connection matrix from a set of allophone contexts. Fig. 6 is an example of connection matrix, where "1" expresses the two allophones are connectable, "0" expresses the two allophones are not connectable.

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

Figure 6: An example of connection matrix

## 4.2 Conversion of the Grammar

We illustrate the grammar conversion with the following simple grammar.

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

Figure 7: An example of the CFG rules and lexical rules

The phones within a word can be converted automatically into the allophones according to a set of allophone contexts. Because of the word juncture problem, in order to express the phones at word boundaries, we introduce allophone rules.

Fig. 8 is the grammar converted from the grammar in Fig. 7.

From the above grammar, a canonical LR table, as shown in Fig. 9, can be generated. (Note: the lookahead symbols are allophones in Fig. 6)

| | |
|---|---|
| (1) S → N BE | (8) a → a2 |
| (2)' N → h a1 h1 a | (9) ch → ch1 |
| (3)' N → ch i2 ch2 i | (10) ch → ch2 |
| (4)' BE → d a | (11) i → i1 |
| (5) h → h1 | (12) i → i2 |
| (6) h → h2 | (13) d → d1 |
| (7) a → a1 | (14) d → d2 |

Figure 8: CFG, lexical and allophone rules

## 4.3 Constraint Propagation Method (CPM)

The LR table in Fig. 9, for the phones at word boundaries, does not include context information yet. In order to incorporate the phoneme context into the phones at word boundaries, we use the connection matrix to delete illegal actions at first, then modify the LR table through constraint propagation.

| | ACTION | | | | | | | | | | | GOTO | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| state | h1 | h2 | a1 | a2 | ch1 | ch2 | i1 | i2 | d1 | d2 | $ | h | a | ch | i | d | N | BE | S |
| 0 | sh5 | sh6* | | | sh2* | sh3 | | | | | | 4 | | 1 | | | 7 | | 25 |
| 1 | | | | | | | | sh8 | | | | | | | | | | | |
| 2* | | | | | | | | re9* | | | | | | | | | | | |
| 3 | | | | | | | | re10 | | | | | | | | | | | |
| 4 | | (c) | sh9 | | | | | | | | | | | | | | | | |
| 5 | | | re5 | | | | | | | | | | | | | | | | |
| 6* | | | re6* | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | sh11 | sh12* | | | | | | 10 | | 13 | |
| 8 | | | | | | sh21 | | | | | | | | | | | | | |
| 9 | sh17 | | | | | | | | | | | | | | | | | | |
| 10 | | | sh15 | sh16* | | | | | | | | | 14 | | | | | | |
| 11 | | | re13 | re13* | | | | | | | | | | | | | | | |
| 12* | | | re14* | re14* | | | | | | (d) | | | | | | | | | |
| 13 | | | | | | | | | | | re1 | | | | | | | | |
| 14 | | | | | | (b) | | | | | re4 | | | | | | | | |
| 15 | | | | | | | | | | | re7 | | | | | | | | |
| 16* | | | | | | | | | | | re8* | | | | | | | | |
| 17 | | | sh19 | sh20* | | | | | | | | | 18 | | | | | | |
| 18 | | | | | | | | | re2 | re2* | (d) | | | | | | | | |
| 19 | | | | | | | | | re7 | re7* | | | | | | | | | |
| 20* | | | | | | | | | re8* | re8* | | | | | | | | | |
| 21 | | | | | | | sh23* | sh24 | | | | | | | 22 | | | | |
| 22 | | | | | | | | | re3 | re3* | | | | | | | | | |
| 23* | | | | | | | (c) | | re11* | re11* | | | | | | | | | |
| 24 | | | | | | | | | re12 | re12* | | | | | | | | | |
| 25 | | | | | | | | | | | acc | | | | | | | | |

Figure 9: Canonical LR (CLR) table generated from rules in Fig. 8

### (4.3.1) Delete the illegal actions and states using connection matrix

#### (a). Check the reduce actions with allophone rules

At first, we use the allophone connection matrix to check all the reduce actions with allophone rule and delete the illegal allophone reduce actions.

For example, re6 with lookahead "a1" in state in Fig. 9 should be deleted, since the connection between "h2" (RHS of rule 6) and "a1" is not allowed (*Connect*[h2,a1] = 0).

#### (b). Check the shift actions

Then, we check all the shift actions whose lookahead symbols are an allophone of the end phone of a word, and delete the illegal shift actions.

For example, 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 sh21 in state 8, we have to shift "i1" and "i2". In Fig. 9, sh23 in state 21 is to shift "i1". *Connect*[ch2, i1]=0 means, however, that shifting "i1" is not allowed. Therefore, sh23 in state 21 should be deleted.

### (4.3.2) Delete the illegal actions and states through constraint propagation

Then we check all other actions by using constraint propagation method.

#### (c). Delete the empty states and the shift actions that transfer to the empty state

An empty state is defined a state whose all actions have been deleted, or all actions that transfer to this state have been deleted. In this case, we should delete this empty state and the shift actions that transfer to this state.

For example, in Fig. 9, sh6 in state 0 with the lookahead symbol "h2" should be deleted, since state 6 is an empty state.

#### (d). Check all the reduce actions

```
for each reduce action R in each entry of LR table {
      search the new state s' after carrying out R;
      if (Action[a, s'] = "error")
         delete R;
      else
         mark R and the action with lookahead a in state s' "found";
  }
where
 a: the lookahead symbol of R.
```

Figure 10: Check all the reduce actions

For example, consider "re2" with lookahead "d2" in state 18, after carrying out "re2", the parser will transfer to to state 7, since Action[17, d2] = "error" (sh12 has been deleted by step (c)), therefore "re2" with lookahead "d2" in state 18 should be deleted.

(e). **Delete the actions that is not marked "found"**

```
for each reduce and shift action A whose predecessors are goto actions
in each entry of LR table {
    if (A is not marked "found") {
        delete A;
    }
```

Figure 11: Delete the actions that is not marked "found"

(f). If no any more action has been deleted, compress the LR table and complete. Otherwise return to step (c).

In Fig. 9, the action that has been deleted by above steps is marked with asterisks (*).

## 4.4 Table Size

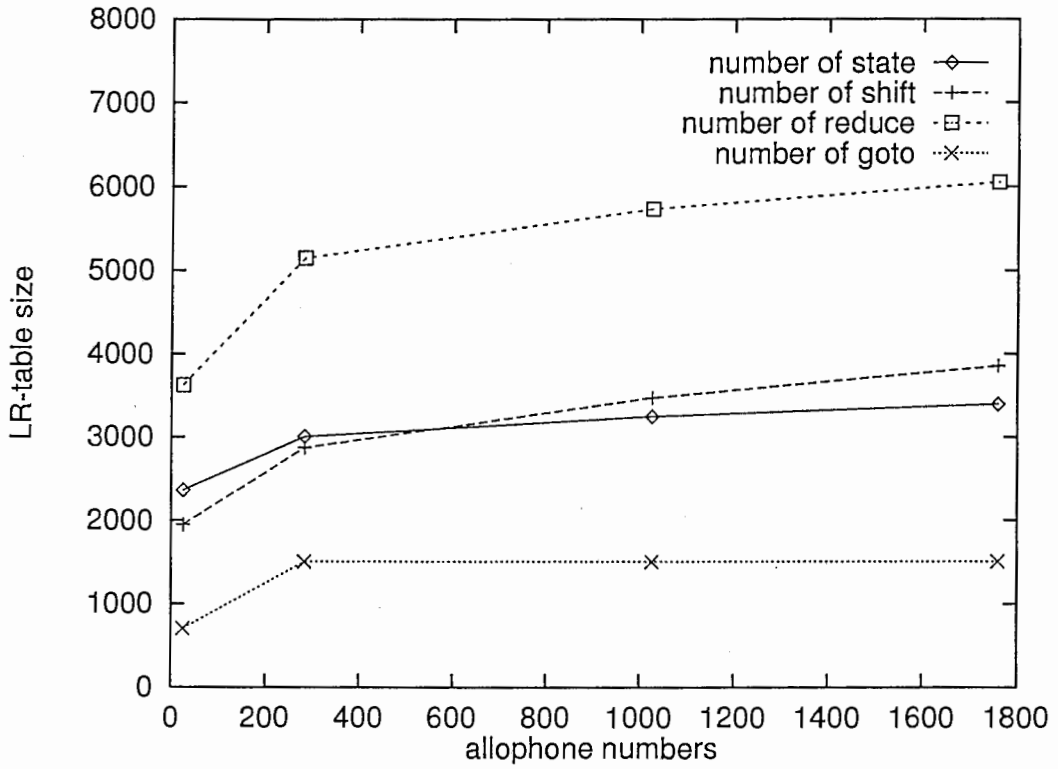



Figure 12: The number of states and actions for mset_phrase grammar

For mset phrase grammar with 979 rules, the number of states and actions of LR table generated by CPM is shown in Fig. 12 (the number of allophones are 283, 1026, 1759). For the sake of comparison, the left of the figure shows the result of LR table in the case of 26 phones.

Table 2 lists a comparison for eset phrase grammar with 2813 rules.

Table 2: The number of states and actions for eset_phrase grammar

| allophone number | state | shift | reduce | goto |
|---|---|---|---|---|
| 26 (*) | 7411 | 6578 | 15428 | 1463 |
| 283 | 9004 | 10686 | 21145 | 4013 |
| 1026 | 9628 | 14020 | 23891 | 4013 |

## 4.5 Incorporation of Connection Constraints into the Generation Process of LR Table

As explained above, we introduced the allophone rules into the phoneme-context-independent grammar at first, then generated an initial allophone-based LR table, and finally deleted all the illegal actions and states.

Although the above method has the advantage of enabling us using already existing LR table generation method to get the initial LR table, the states and actions of initial LR table often explode as the number of grammar rules and allophone models increases. For example, for a grammar with 312 CFG and lexical rules, in the case of 1024 allophones, the number of the states of initial LR table is 11157 (the number of states after CPM is 1231).

In order to rectify this situation, we can incorporate the connection constraints between adjacent allophones (step (a) and (b)) into the generation process of LR table.

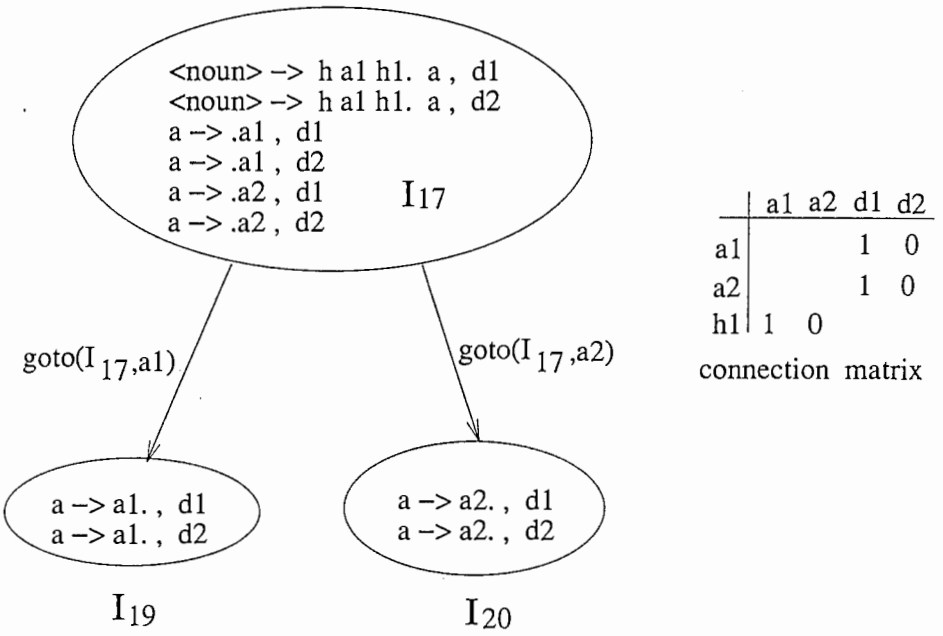



Figure 13: Incorporation of connection constraints into the generation process of LR table

For example, consider the item set $I_{17}$ in Fig. 13,

- Since $Connect[h1, a2] = 0$, the following two items of item set $I_{17}$

  a → .a2, d1

  a → .a2, d2

can be deleted, this corresponds to step (b) of CPM, and because the above two items were deleted, the item set $I_{20}$ will vanish automatically.

- Since $Connect[a1, d2] = 0$, so the item

  a → .a1, d2

  should be deleted, this corresponds to step (a) of CPM.

By the above two steps, the three item sets in Fig. 13 will decrease to two item sets as shown in Fig. 14.

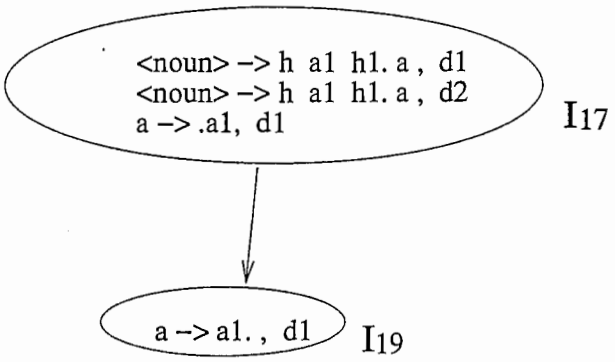



Figure 14: Item sets after using connection constraints

The size of initial allophone-based LR table can be reduced greatly through this method.

# 5 Parser Algorithm

As phoneme context has been compiled in the LR table in advance, the LR parsing algorithm is principally the same as Tomita's GLR parsing algorithm. There is only little change about GLR parsing algorithm with the allophone-based LR table.

Consider the following grammar and the LR table obtained by CPM.

Grammar

(1) <s> –> <n> <p>
(2) <n> –> a n1 i
(3) <n> –> a n2 e
(4) <p> –> g a
(5) <i> –> i1
(6) <e> –> e1
(7) <g> –> g1
(8) <g> –> g2

Connection matrix

| | g1 | g2 |
|---|---|---|
| i1 | 1 | 0 |
| e1 | 0 | 1 |

LR table

| | i1 | e1 | g1 | g2 | i | e | <n> |
|---|---|---|---|---|---|---|---|
| 0 | | | | | | | 11 |
| 5 | sh7 | | | | 9 | | |
| 6 | | sh8 | | | | 10 | |
| 7 | | | re5 | | | | |
| 8 | | | | re6 | | | |
| 9 | | | re2 | | | | |
| 10 | | | | re3 | | | |
| 11 | | | sh12 | sh13 | | | |

In state 7, there is only one allophone reduce action with lookahead "g1" for the end phone "i" of word "a n1 i". After carrying out this reduce, the parser transfers to state 9. In state 9, another reduce action of word "'a n1 i" with lookahead "g1" is carried out, and goto state 11. However, in state 11, there are two shift actions with lookahead "g1" and "g2". Since "g2" can not succeed "i1", so for word "a n1 i", sh13 is illegal. But we can not delete this action from the table, because after the reduce of word "a n2 e", the parser will transfer to this state too, and "g2" can succeed "e1".

To avoid this useless prediction, we have to modify the GLR parsing algorithm. For the reduce action with allophone rule, its lookahead symbol should be stored and taken to the new state after reduce, and at the new state, only the actions with the same lookahead are carried out.

In the above example, if the path is "a n1 i", since the lookahead symbol of reduce action "i → i1" is "g1", we take it to the state 11, and in state 11, only sh12 with lookahead "g1" is carried out.

The complete algorithm is summarized below:

## [LR Parsing Algorithm:]

As same as SSS-LR GLR parser algorithm, a data structure (named cell) with information about one possible parse is used.

The following information is kept in the cell:

- LR stack, with information for parsing control.

- Prediction allophone set P, which includes the next allophones the parser will predict, is used to remove the ambiguity of allophone predictions when the actions are transfered from a goto action.
- Probability array, which includes end point candidates and their probabilities.

1. Initialization
   create a new cell, push the LR initial state 0 on the top of LR stack of C.

2. Ramification of cells
   Construct a set.
   $S = \{(C, s, a, x) \mid C, a, x$ (C is a cell & C is not accepted
   & s is the top state of C & ACTION[s, a] = x & x $\neq$ error.)$\}$
   for each element $(C, s, a, x) \in S$, do the following. If set S is empty, parser is completed.

3. if x = "shift $s'$",
   - if state s is transfered from a shift action, verify the existence of allophone "a", and let the prediction allophone set P be an empty set.
   - if state s is transfered from a goto action and the input "a" is included in the prediction set P, verify the existence of allophone "a", otherwise the cell C is abandoned.

4. if x = "reduce n",
   - if rule n is an allophone rule, do this reduce action, and add the allophone "a" into P.
   - if rule n is not an allophone rule and "a" is included in P, do this reduce action. Otherwise, the cell C is abandoned.

5. if x = accept, and the probability of cell C exceeds a certain threshold, cell C is accepted. If not, cell C is abandoned.

6. Return to 2.

# 6 Speech Recognition Experiments

## 6.1 Speech Data

The recognition experiments were carried out using 345 phrases uttered by one professional announcer (MAU in the ATR database).

The speech was sampled at 12 kHz, quantized to 16 bits, preemphasized by $(1-0.98z^{-1})$, and windowed using a 20 msec Hamming window with a 5 msec shift, 34 coefficients which consists of log-power, delta log-power, 16-channel cepstrum coefficients, 16-channel delta cepstrum coefficients were used as the feature parameters, a diagonal-covariance single Gaussian distribution was used as an output probability density distribution of each state. Isolated 2620 Japanese words (even words of ATR 5240-word database) were used for training data. Allophone model (called HMnet) is generated by a SSS algorithm [9]. The number of states for the HMnet are 200, 400 and 600, which corresponds to 283, 1026 and 1759 allophones, respectively.

## 6.2 Grammars

Two kinds of phrase grammar for international conference secretarial service (mset_phrase and eset_phrase) were used. Grammar size and complexity are shown in Table 3.

Table 3: Size and perplexity of grammar

| Grammar | mset_phrase | eset_phrase |
|---|---|---|
| Rule | 979 | 2813 |
| Vocabulary | 456 | 1588 |
| Perplexity (phone) | 2.66 | 3.43 |

## 6.3 Recognition Results

In the following experiments, for parser level realization, phoneme-context-independent and phoneme-context-dependent duration models were used. For table level realization, only phoneme-context-independent duration model was used, the training of duration model based allophones is difficult.

The comparisons of recognition rate, CPU time, number of allophone verifications and accepted phrase candidates were performed.

### (6.3.1) Results for mset_phrase grammar

#### 1. Results with phoneme-context-independent duration model

##### 1). Recognition rate

The recognition rates for parser level realization with SLR and CLR table are listed in Table 4. The number of allophone models is 283, 1026 and 1759. The results of phoneme-context-independent model (26 phones) are given too.

It is easy to see that when beam width is small, canonical LR table gives slightly better recognition rates. This indicates that the CLR table gives more precise allophone predictions than the SLR table.

For 1026 allophone models, the better recognition performance has been obtained. When the number of allophone models increased to 1759, the recognition rate remained about the same as that of 1026 allophone models. This implies that 1026 allophone models can model the coarticulatory effects in continuous speech better.

Since the duration model is phoneme-context-independent, although use the acoustic phoneme-context-dependent model, the recognition rates are almost the same as that of phoneme-context-independent model. In the case of 283 allophones, the performance is degraded comparing to that of phoneme-context-independent model.

Because the table level and parser level realization is in principle equivalent, the recognition rates of table level realization based on CPM are exactly the same as that of parser level realization using CLR table.

Table 4: Recognition rates for mset_phrase

| Allophone number | Table type | beam=20 | | beam=50 | | beam=100 | | beam = 250 | |
|---|---|---|---|---|---|---|---|---|---|
| | | top 1 | top 5 | top 1 | top 5 | top 1 | top 5 | top 1 | top 5 |
| 26 (*) | CLR | 90.72 | 94.20 | 92.64 | 96.81 | 93.33 | 97.97 | 94.78 | 99.13 |
| 283 | SLR | 86.38 | 92.75 | 88.70 | 95.65 | 90.14 | 97.10 | 91.68 | 98.84 |
| | CLR | 86.96 | 93.33 | 88.70 | 95.65 | 90.14 | 97.10 | 91.68 | 98.84 |
| 1026 | SLR | 92.46 | 96.81 | 93.33 | 97.97 | 93.91 | 98.55 | 94.78 | 99.13 |
| | CLR | 93.04 | 97.39 | 93.62 | 98.26 | 93.91 | 98.55 | 94.78 | 99.42 |
| 1759 | SLR | 92.17 | 97.10 | 93.04 | 98.55 | 93.62 | 99.13 | 93.91 | 99.42 |
| | CLR | 92.75 | 97.68 | 93.04 | 98.55 | 93.62 | 99.13 | 93.91 | 99.42 |

**2). CPU time**

The average CPU time (measured in an HP735) for each utterance is shown in Fig. 15. Since 90% of CPU time is spent on the verification of allophones, the CPU time is almost the same for table level and parser level realization.

**3). Average allophone verifications**

The average allophone verifications for each phrase utterance are listed in Table 5.

Table 5: Average number of allophone verifications for mset_phrase

| Allophone number | Realization | beam=20 | beam=50 | beam=100 | beam=250 |
|---|---|---|---|---|---|
| 283 | SLR parser level | 720 | 1648 | 3009 | 6854 |
| | CLR parser level | 696 | 1571 | 2832 | 6370 |
| | CLR table level | 518 | 1235 | 2275 | 5229 |
| 1026 | SLR parser level | 613 | 1431 | 2634 | 6370 |
| | CLR parser level | 604 | 1402 | 2568 | 6028 |
| | CLR table level | 513 | 1240 | 2305 | 5314 |
| 1759 | SLR parser level | 594 | 1374 | 2518 | 5800 |
| | CLR parser level | 587 | 1353 | 2458 | 5653 |
| | CLR table level | 516 | 1246 | 2306 | 5347 |

It can be seen that among the three realizations, the parser level realization using SLR table needs more verifications under the same conditions, the table level realization, because

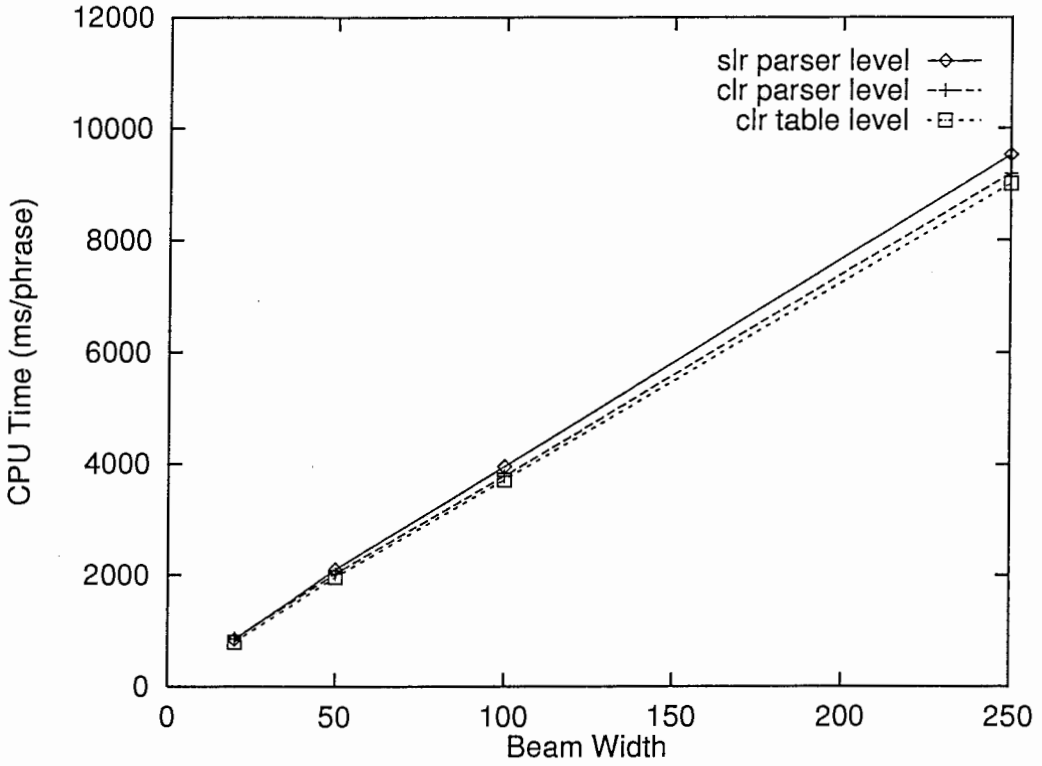



Figure 15: Average CPU time (mset_phrase, allophone numbers=1759)

the phoneme context is compiled in advance, needs the less verifications.

**4). Average number of accepted phrase candidates**

The average number of accepted phrase candidates for each phrase utterance are listed in Table 6. Under the same conditions, CLR table provides more phrase candidates than SLR table.

Table 6: Average number of accepted phrase candidates for mset_phrase

| Allophone number | Table type | beam=20 | beam=50 | beam=100 | beam=250 |
|---|---|---|---|---|---|
| 283 | SLR | 33 | 90 | 177 | 421 |
| | CLR | 34 | 96 | 190 | 462 |
| 1026 | SLR | 30 | 80 | 155 | 366 |
| | CLR | 32 | 89 | 175 | 425 |
| 1759 | SLR | 29 | 78 | 151 | 358 |
| | CLR | 32 | 87 | 171 | 415 |

**2. Results with phoneme-context-dependent duration model**

The experiments for parser level realization using two types of LR table (SLR and CLR) were carried out.

Table 7 shows the recognition rates for parser level realization.

When the beam width is small, similar to the phoneme-context-independent case, canonical LR table gives slightly better recognition rates.

Compared with the results of using phoneme-context-independent duration model, for the case of 283 allophones, the recognition rate improved about 1.38% (beam=250), but as the number of allophone models increases, the recognition rate is almost the same as that of phoneme-context-independent duration model.

Table 7: Recognition rate for phoneme-context-dependent duration model

| Allophone number | Table type | beam=20 | | beam=50 | | beam=100 | | beam = 250 | |
|---|---|---|---|---|---|---|---|---|---|
| | | top 1 | top 5 | top 1 | top 5 | top 1 | top 5 | top 1 | top 5 |
| 283 | SLR | 88.99 | 94.49 | 91.01 | 97.10 | 92.75 | 98.84 | 93.04 | 99.13 |
| | CLR | 89.28 | 94.78 | 91.01 | 97.10 | 92.75 | 98.84 | 93.04 | 99.13 |
| 1026 | SLR | 93.04 | 97.39 | 94.20 | 98.84 | 94.20 | 98.84 | 94.78 | 99.42 |
| | CLR | 93.62 | 97.97 | 94.20 | 98.84 | 94.20 | 98.84 | 94.78 | 99.42 |
| 1759 | SLR | 92.46 | 97.39 | 93.33 | 98.84 | 93.62 | 99.13 | 93.91 | 99.42 |
| | CLR | 92.75 | 97.68 | 93.33 | 98.84 | 93.62 | 99.13 | 93.91 | 99.42 |

### 3. Recognition rates if including "silence" allophone models

So far phone context has not been considered for "silence" model. Only one allophone is assumed in the above allophone models for "silence". In practice, the acoustic character of "silence" changes according to the preceding and succeeding phones.

Adding the "silence" allophone models into the above three allophone models, the numbers of allophones become 309 (including 16 "silence" allophones), 1062 (including 36 "silence" allophones) and 1795 (including 36 "silence" allophones) respectively.

Table 8 shows the recognition rates for parser level realization in the case of including the "silence" allophones.

It can be seen that when the beam width is small (beam=20, 50), the recognition rates are slightly worse. As the beam width increases (beam=100, 250), the recognition rates are almost the same as that of only one "silence" allophone. This indicated that, in the case of phrase recognition, since "silence" only appears at the beginning or end of a phrase, its acoustic character is very different from other phones, and can be characterized by only one model.

Table 8: Recognition rate when the "silence" allophones are included

| Allophone number | Table type | beam=20 | | beam=50 | | beam=100 | | beam = 250 | |
|---|---|---|---|---|---|---|---|---|---|
| | | top 1 | top 5 | top 1 | top 5 | top 1 | top 5 | top 1 | top 5 |
| 309 | SLR | 87.54 | 93.33 | 91.59 | 97.97 | 92.46 | 98.84 | 93.04 | 99.42 |
| | CLR | 87.83 | 93.62 | 91.59 | 97.97 | 92.46 | 98.84 | 93.04 | 99.42 |
| 1062 | SLR | 89.86 | 94.49 | 93.91 | 98.55 | 94.20 | 98.84 | 94.78 | 99.42 |
| | CLR | 90.43 | 95.07 | 93.91 | 98.55 | 94.20 | 98.84 | 94.78 | 99.42 |
| 1795 | SLR | 90.14 | 94.78 | 93.62 | 98.84 | 93.62 | 98.84 | 94.20 | 99.42 |
| | CLR | 90.72 | 95.36 | 93.62 | 98.84 | 93.62 | 98.84 | 94.20 | 99.42 |

### (6.3.2) Results for eset_phrase grammar

#### 1. Results with phoneme-context-independent duration model

**1). Recognition rate**

Table 9 shows the recognition rate of parser level realization.

As in the case of mset_phrase grammar, when beam width is small, canonical LR table gives slightly better recognition rates.

From Table 9, the same conclusions as for mset_phrase hold.

Table 9: Recognition rates for eset_phrase

| Allophone number | Table type | beam=20 | | beam=50 | | beam=100 | | beam = 250 | |
|---|---|---|---|---|---|---|---|---|---|
| | | top 1 | top 5 | top 1 | top 5 | top 1 | top 5 | top 1 | top 5 |
| 26(*) | CLR | 71.30 | 75.07 | 82.90 | 90.43 | 85.51 | 94.20 | 87.83 | 97.10 |
| 283 | SLR | 68.70 | 75.07 | 79.13 | 87.25 | 82.32 | 92.17 | 84.93 | 95.65 |
| | CLR | 69.57 | 75.94 | 79.71 | 88.41 | 82.32 | 92.17 | 84.93 | 95.65 |
| 1026 | SLR | 75.94 | 82.90 | 84.93 | 92.75 | 88.41 | 96.23 | 88.99 | 97.97 |
| | CLR | 77.39 | 84.35 | 85.51 | 93.33 | 88.41 | 96.52 | 88.99 | 97.97 |
| 1759 | SLR | 78.55 | 84.35 | 85.51 | 92.75 | 88.12 | 96.23 | 89.28 | 98.26 |
| | CLR | 80.29 | 85.80 | 86.38 | 93.91 | 88.12 | 96.52 | 89.28 | 98.26 |

**2). CPU time**

The average CPU time is shown in Fig. 16.

**3). Average verifications for each phrase utterance**

The average allophone verifications for each phrase utterance are listed in Table 10.

Table 10: Average phonetic verifications for eset_phrase

| Allophone number | Realization | beam=20 | beam=50 | beam=100 | beam=250 |
|---|---|---|---|---|---|
| 283 | SLR parser level | 1145 | 2552 | 4315 | 9639 |
| | CLR parser level | 1106 | 2444 | 4105 | 9126 |
| | CLR table level | 722 | 1697 | 2993 | 6903 |
| 1026 | SLR parser level | 898 | 2054 | 3654 | 8134 |
| | CLR parser level | 878 | 1998 | 3536 | 7840 |
| | CLR table level | 699 | 1661 | 3070 | 6893 |
| 1759 | SLR parser level | 823 | 1880 | 3381 | 7567 |
| | CLR parser level | 808 | 1839 | 3286 | 7334 |

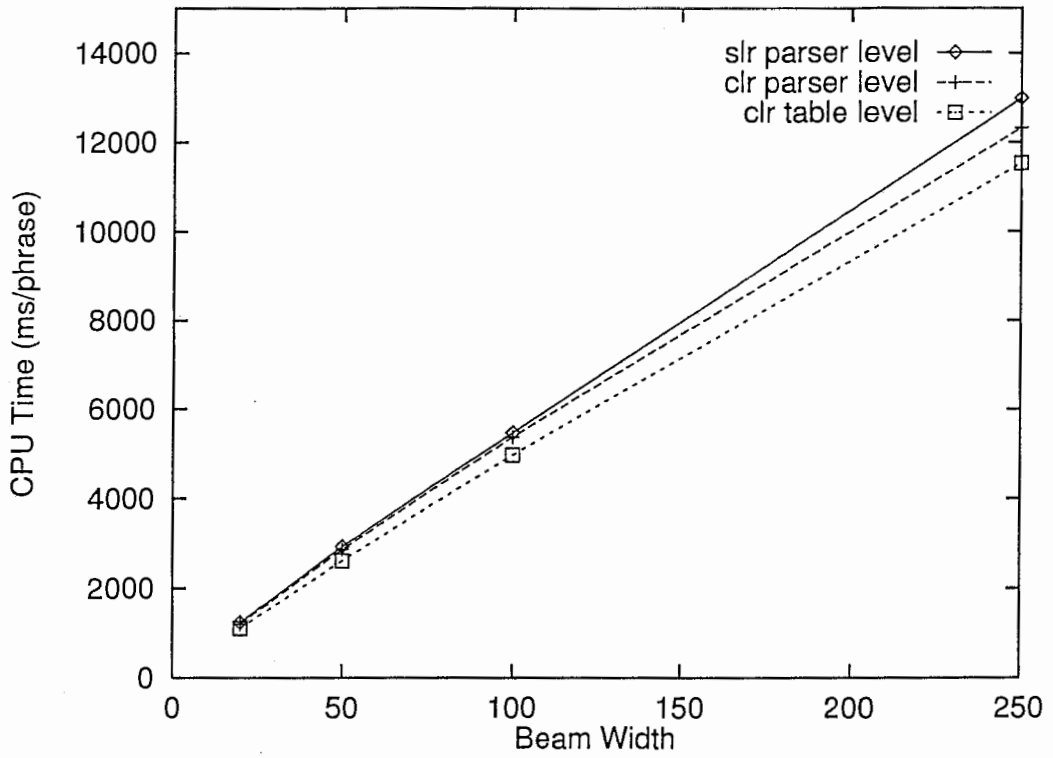



Figure 16: Average CPU time (eset_phrase, allophone numbers=1026)

**4). Average number of accepted phrase candidates for each phrase utterance**

Table 11 lists the average number of accepted phrase candidates for each phrase utterance.

Table 11: Average number of accepted phrase candidates for eset_phrase

| Allophone number | Table type | beam=20 | beam=50 | beam=100 | beam=250 |
|---|---|---|---|---|---|
| 283 | SLR | 26 | 73 | 152 | 386 |
| | CLR | 32 | 88 | 181 | 454 |
| 1026 | SLR | 22 | 63 | 133 | 340 |
| | CLR | 28 | 81 | 165 | 417 |
| 1759 | SLR | 21 | 60 | 128 | 330 |
| | CLR | 28 | 78 | 160 | 406 |

**2. Results with phoneme-context-dependent duration model**

Table 12 shows the recognition rate of parser level realization

Compared with the result of phoneme-context-independent duration model, In the case of 283 allophones, the recognition rate improved about 2.9% (beam=250), in the case of 1026 allophones, the recognition rate improved about 2.02%, and for 1759 allophones, the recognition rate is about the same.

Table 12: Recognition rate for phoneme-context-dependent duration model

| Allophone number | Table type | beam=20 | | beam=50 | | beam=100 | | beam = 250 | |
|---|---|---|---|---|---|---|---|---|---|
| | | top 1 | top 5 | top 1 | top 5 | top 1 | top 5 | top 1 | top 5 |
| 283 | SLR | 73.33 | 79.71 | 83.48 | 90.72 | 85.80 | 95.07 | 87.83 | 97.10 |
| | CLR | 73.33 | 79.71 | 84.06 | 91.30 | 86.09 | 95.65 | 87.83 | 97.39 |
| 1026 | SLR | 80.29 | 86.38 | 88.12 | 95.07 | 90.72 | 97.68 | 91.01 | 99.13 |
| | CLR | 80.29 | 86.38 | 88.70 | 95.94 | 90.43 | 98.26 | 90.72 | 99.13 |
| 1759 | SLR | 81.74 | 87.25 | 86.96 | 94.49 | 89.28 | 97.68 | 89.28 | 98.84 |
| | CLR | 82.03 | 88.12 | 87.54 | 95.36 | 89.28 | 97.68 | 89.28 | 98.84 |

# 7 Discussions and Conclusions

In this report, we applied the phoneme-context-dependent LR table based on CPM method to SSS-LR continuous speech recognition system.

We summarize our work below.

- Canonical LR table can provide us more precise allophone predictions than SLR table. The recognition performance was compared by recognition experiments. For phrase recognition tasks, using CLR table instead SLR table, the average allophone verifications and CPU time decreased about 2.5% - 10%, and the average phrase candidates increased about 15% under the same conditions. When beam width is 20, the error rate decreased by 8% for eset_phrase grammar.

- The allophone-based LR table is practical for large vocabulary continuous speech recognition. By using CPM method, the table size can be reduced greatly, and at the same time the word juncture problem can solved better.

- As phoneme context has been compiled into the LR table beforehand, there is only little change about the GLR parsing algorithm. The parsing algorithm of table level realization is simpler and clearer than that of parser level realization. This is beneficial for further development of SSS-LR system.

- Compared with the parser level realization, the average allophone verifications and CPU time decreased about 5%, since duplicate verifications are avoided.

One problem about CPM method is that generating an allophone-based LR table is time-consuming for a large grammar compared with phoneme-context-independent case. But for a given grammar, the LR table can be generated in advance.

The future work will include the following:

- Incorporating the phoneme-context-dependent LR table into continuous sentence recognition system.

- Integration of phoneme-context-dependent acoustic model and phoneme-context-dependent duration model.

- Processing of unknown word.

  In speech recognition, to solve the problem of unknown word, it is often required to add some rules or words into the grammar. Regenerating the allophone-based LR table is time-consuming, it is useful to incorporate the unknown word into the already existing LR table through considering the relationship between the unknown word and the existing grammar.

- Application to stochastic CFG grammar and constructing allophone-based stochastic LR table.

  Since in CPM, all the allophones have been introduced the grammar through allophone rules, it is possible to construct an allophone-based stochastic LR table with the same method as in the phoneme context independent case. If using parser level realization, the probability of each action would have to computed dynamically during recognition process, and this would make the GLR parser much more complex and inefficient.

## Acknowledgments

The authors are grateful to Dr. Yamazaki, the president of ATR Interpreting Telecommunications Research Laboratories for giving us this research opportunity, and thanks Prof. Tanaka, Prof. Tokunaga, Tokyo Institute of Technology, Mr. Morimoto, the head of Department 4 of ATR-ITL for their guidance and support. We would also like to thank all the members of ATR-ITL and Tanaka Laboratory, Tokyo Institute of Technology.

We are especially grateful to Dr. Suresh (Tokyo Institute of Technology), who wrote the Prolog program for generating canonical LR table.

## 参考文献

[1] Aho, A.V., Sethi, R. and Ullman, J.D. *Compilers: Principles, Techniques, and Tools.* Massachusetts, Addison-Wesley, 1986.

[2] Hayamizu, S., Lee, K.F. and Hon, H.W. *Description of Acoustic Variations by Tree-Based Phone Modeling.* Proc. ICSLP90, pp. 705-708, 1990.

[3] 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. ICASSP92, pp. 21-24, 1992.

[4] Kita, K., Kawabata, T. and Saito, H. *HMM Continuous Speech Recognition Using Predictive LR Parsing.* Proc. ICASSP89, 1989.

[5] Lari, K. and Young, S.J. *The Estimation of Stochastic Context-Free Grammars Using the Inside-Outside Algorithm.* Computer Speech and Language, No. 4, pp. 35-56, 1990.

[6] Lee, K.F. *Automatic Speech Recognition: The Development of the SPHINX System.* Kluwer Academic Publishers, Norwell, MA, 1989.

[7] Nagai, A., Takami, J., Sagayama, S. *The SSS-LR Continuous Speech Recognition System: Integrating SSS-derived Allophone Models and a Phoneme-Context-Dependent LR Parser.* IEICE Technical Reports (信学技報), SP92-33, 1992.

[8] 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, January, 1993.

[9] Nagai, A., Takami, J., Sagayama, S., and Singer, H. *Continuous Speech Recognition Integrating Hidden Markov Networks and a Generalized LR Parser.* (in Japanese) Transactions of the Institute of Electronics, Information and Communication Engineers, **J77-D-II**(1):9–19, 1994.

[10] Nagai, A., Singer, H. *SSS-LR* 連続音声認識システム： *User's Guide.* ATR Technical Report, TR-I-0339, March, 1993.

[11] Schwartz, R., Chow, Y., Kimball, O., Roucos, S. Krasner, M. and Makhoul, J. *Context Dependent Modeling for Acoustic Phonetic Recognition of Continuous Speech.* Proc. ICASSP85, 1985.

[12] Takami, J. and Sagayama, S. *A Successive State Splitting Algorithm for Efficient Allophone Modeling.* Proc. ICASSP92, Vol. **1**, pp. 573-576, San Francisco, March 1992.

[13] Tanaka, H., Tokunaga, T. and Aizawa, M. *Integration of Morphological and Syntactic Analysis Based on LR Parsing Algorithm.* International Workshop on Parsing Technologies, Tilburg, pp. 101-109, 1993.

[14] Tanaka, H., Li, H. and Tokunaga, T. *Incorporation of Phoneme-Context-Dependence into LR Table through Constraint Propagation Method.* will appear at AAAI-94 Workshop on the Integration of Natural Language and Speech Processing, Seattle, 1994.

[15] Tomita, M. *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems.* Kluwer Academic Publishers, p. 201, 1986.

[16] Wright, J.H. *LR Parsing of Probabilistic Grammars with Input Uncertainty for Speech Recognition.* Computer Speech and Language, Vol. **4**, pp.297-323, 1990.

# 付録 A Program for Generating Allophone-based Canonical LR Table

### 1. Allophone context map file

To generate allophone-based canonical LR table, allophone context map file is necessary. The format of allophone context map file is as follows:

allophonename centerphone=leftphone_rightphone

For example, context map of allophone "a1" is:

a1 a=e_r

This means that the center phone of allophone "a1" is "a", the left phone is "e" and the right phone is "r".

### 2. Conversion of grammar

With program "grchange", phoneme-context-independent grammar can be changed into a phoneme-context-dependent grammar.

usage: grchange
[-a allophone context map file]
input: phoneme-context-independent grammar (standard input)
output: allophone-based grammar (standard output)

### 3. Generation of allophone-based canonical LR table

usage: clr
[-a allophone context map file]
[-v debug option]
input: allophone-based grammar (standard input)
output: allophone-based canonical LR table (standard output)

### 4. Modification of allophone-based canonical LR table by using CPM

usage: cpm
input: output allophone-based grammar of "clr" (standard input)
output: modified allophone-based canonical LR table (standard output)

### 5. Conversion of phrase or sentence

Usually, we use GLR parser to check whether the LR table is correct or not. In this case, allophone-based test phrases or sentences are necessary. Program "chwrd" changes the phoneme-context-independent phrase or sentence into allophone-based phrase or sentence.

usage: chwrd
[-a allophone context map file]
input: phoneme-context-independent phrase/sentence (standard input)
output: allophone-based phrase/sentence (standard output)