Vol. 36 No. 5 Transactions of Information Processing Society of Japan May 1995

Regular Paper

One-Pass Search Algorithm for Continuous Speech Recognition Using Generalized LR Parsing: A CFG-Driven, Frame-Synchronous HMM-Based Approach

KENJI KITA,† YONEO YANO † and TSUYOSHI MORIMOTO ††

In this paper, we present a novel continuous speech recognition algorithm that integrates three major technologies: (1) hidden Markov models for speech, (2) a generalized LR parser for handling context-free grammar (CFG) constraints, and (3) the one-pass search algorithm for efficient searching. We also introduce three techniques that used in the development of the algorithm: (1) LR path-merging, (2) the use of a shared tree-structured stack, and (3) LR-parser-based dynamic network generation. By means of the proposed algorithm, an optimal hypothesis can be found efficiently for a given speech signal according to a specified CFG in a frame-synchronous process. We implemented an experimental Japanese speech recognition system based on the proposed algorithm, using discrete-type context-independent HMMs without duration control. The system attained a recognition accuracy of 84.1%-88.1%, depending on the beam width. We also experimentally compared our algorithm with the following two methods: (1) the one-pass search algorithm using the finite-state approximation for a CFG, and (2) the HMM-LR algorithm. The experiments showed that the proposed algorithm attained higher accuracy when the beam width was small.

1. Introduction

In this paper, we will formulate a novel continuous speech recognition algorithm that integrates three major technologies: (1) hidden Markov models for speech,[6] (2) a generalized LR parser for handling context-free grammar (CFG) constraints,[20),21)] and (3) the one-pass search algorithm for efficient searching.[4),11),15)] By means of the algorithm, an optimal hypothesis can be found efficiently for a given speech signal according to a specified CFG in a frame-synchronous process.

The algorithm amalgamates and extends three techniques already proposed: (1) grammar node path-merging in a syntax-directed one-pass search, (2) the use of a tree-structured representation of an LR parser's state stack, and (3) dynamic grammar network generation and the predictive use of an LR parsing table. In the first technique, syntactic constraints are introduced into the one-pass search algorithm by means of grammar node path-merging, which searches for the best path reaching each grammar state in each time frame. When the syntactic constraints are described by a finite-state automaton (FSA), a grammar state corresponds to a node in the FSA. In the case of context-free grammar (CFG), however, path-merging is not straightforward. Kai et al. introduced path-merging for CFGs based on Earley's parsing algorithm, but their method requires the use of backtracking to identify a grammar node. In this paper, we will introduce *LR path-merging*, which does not require backtracking. The purpose of LR path-merging is to represent acceptable sentences in a state transition network by considering an LR state sequence as a grammar state. This network is then used to guide the one-pass search in order to select the path that best matches the input speech. In the second technique, efficient representation of an LR parser's stack is achieved by using a tree-structured or graph-structured stack. Our algorithm uses a tree-structured stack, in which the bottom of the stack corresponds to the root node of the tree. This type of stack is an efficient implementation of LR path-merging, because a tree node represents a stack content. The same kind of stack has been used in other LR-parser-based speech recognition systems.[7),13)] The third technique, dynamic grammar network generation, is for incrementally generating the state transition network needed by the speech recognizer.[14)] This technique is used in our algorithm in combination with the predictive use of an LR parsing table.[9)]

This paper is organized as follows. Section 2

† Faculty of Engineering, Tokushima University
†† ATR Interpreting Telecommunications Research Laboratories

describes our algorithm, emphasizing three key techniques: (1) LR path-merging, (2) the use of a shared tree-structured stack, and (3) LR-parser-based dynamic network generation. In Section 3, our algorithm is evaluated through recognition experiments. In Section 4, our algorithm is experimentally compared with other algorithms, which include the one-pass search algorithm using the finite-state approximation for a CFG, and the HMM-LR algorithm. Finally, Section 5 presents our conclusions.

2. LR-Parser-Driven One-Pass Search Algorithm

This section develops a novel continuous speech recognition algorithm that integrates (1) hidden Markov models, (2) a generalized LR parser, and (3) the one-pass search algorithm. The key ideas of this algorithm include (1) LR path-merging, (2) shared tree-structured stack, and (3) LR-parser-based dynamic network generation. First, we will describe these key ideas one by one, and after that we will present the recognition algorithm.

2.1 LR Path-Merging

In the FSA-based one-pass search, recognition corresponds to finding the optimal path through the complete FSA state network. Thus, recognition paths (hypotheses) reaching the same FSA state are merged; that is, all non-optimal paths reaching that node are eliminated.

S	→	NP	VP	
NP	→	DET	NOUN	
NP	→	DET	ADJ	NOUN
VP	→	VERB	NP	
DET	→	a		
ADJ	→	young		
NOUN	→	man		
NOUN	→	woman		
VERB	→	loves		

Fig. 1 Example of a CFG.

Since we are concerned with the LR-parser-driven one-pass search, we perform path-merging according to the LR state sequence (the stack content of the LR parser). This merging is supported by the fact that paths with the same LR state sequence have an identical grammatical function with respect to the subsequent word strings.

Let us give a simple example. We consider a CFG in **Fig. 1** and its LR parsing table in **Fig. 2**. After recognizing "a man", the LR parser will have the following stack: "0 3" (the leftmost item is the bottom of the stack). Once again, for the partial sentence "a young man", the stack will be "0 3". These two partial sentences have different word strings in their surface form, but are dominated by the same nonterminal NP and thus have the same stack content. Accordingly, we can perform path-merging for these two partial parses. **Figure 3** shows the LR path-merging for the two partial parses.

2.2 Shared Tree-Structured Stack

The simplest way to implement LR path-merging is to associate each recognition path with an LR state stack, and compare two stacks by examining their elements one by one. This method is, however, both memory-consuming and time-consuming.

In order to implement LR path-merging efficiently, we represent multiple stacks as a single tree-structured stack, which is shared among all recognition paths. **Figure 4** shows as an example the tree-structured stack that corresponds to the stacks in Fig. 3. In the shared tree-structured stack, the bottom of the stack corresponds to the root of the tree. Each node has a pointer to its parent node, and thus if a node is given, the stack content is uniquely

	a	young	man	woman	loves	$	S	NP	VP	DET	ADJ	NOUN	VERB
0	s1						4	3		2			
1		r5	r5	r5									
2		s8	s5	s6							9	7	
3					s10				12				11
4						acc							
5					r7	r7							
6					r8	r8							
7					r2	r2							
8			r6	r6									
9			s5	s6								13	
10	r9												
11	s1							14		2			
12						r1							
13					r3	r3							
14						r4							

Fig. 2 LR parsing table for the CFG in Fig. 1.

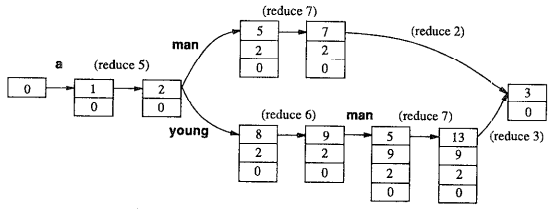


Fig. 3 Example of LR path-merging.

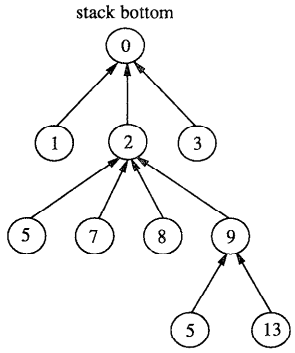


Fig. 4 Example of a shared tree-structured stack.

determined.

2.3 Dynamic Network Generation

The concept of *dynamic grammar network generation*[14] has become more widely known in the past few years. The idea is to incrementally generate a state transition network from a complicated grammar in order to provide a tractable recognition search space. The realization of dynamic grammar network generation is implementation-dependent, and various methods have been reported.[8),16)]

Besides efficiency, generalized LR parsing has another advantage: it can easily be used as a language source model for symbol prediction/generation.[9)] More specifically, for any partial sentence, the possible subsequent terminal symbols can be predicted by referring to a parsing table.

This predictive property makes it possible to generate the state transition network dynamically and efficiently. The initial network contains an initial node whose label is 0, which is the initial state of the LR parser. The initial LR state expects some terminal symbols to be shifted, and therefore these symbols and the next states are added to the network. This process is iterated during recognition.

The following is a specification of the dynamic network generation algorithm based on generalized LR parsing. An example of a network generated by this algorithm is shown

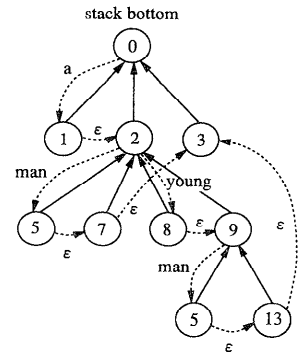


Fig. 5 Example of a dynamically generated network.

in **Fig. 5**, where dotted arrows indicate state transitions. The algorithm generates a state transition network with ε-transitions, but ε-transitions can be easily eliminated.[1)] We denote a state transition of the ε-free network as $\hat{\delta}(v, x)$.

[Notations]

V Set of active nodes.

R Set of 2-tuples $\langle v, n\rangle$, where v is a node and n indicates the n-th production. The 2-tuple $\langle v, n\rangle$ means that the n-th production is to be applied on the node v.

S Set of 3-tuples $\langle v, x, s\rangle$, where v is a node, x is a terminal symbol, and s is a state number. The 3-tuple $\langle v, x, s\rangle$ means that "shift s" by the symbol x is to be applied on the node v.

ACTION$[s, x]$ Action table of the LR parser, where s is an LR state number and x is a terminal symbol.

GOTO$[s, X]$ Goto table of the LR parser, where s is an LR state number and X is a nonterminal symbol.

STATE(v) Function that returns the label (an LR state number) of the node v.

LHS(n) Function that returns symbol in the left-hand side of the n-th production.

LENGTH(n) Function that returns

	the length of the right-hand side of the n-th production.
ANCESTOR(v, m)	Function that returns an ancestor node whose distance from v is m.

[Dynamic Network Generation Algorithm]

GENERATE-NETWORK

```
R⇐φ
S⇐φ
repeat
  if V≠φ
    remove one node v∈V
    for each terminal symbol x
      if ACTION[STATE(v), x]="shift s'"
        S⇐S∪⟨v, x, s'⟩
      else if ACTION[STATE(v), x]
                                ="reduce n"
        R⇐R∪⟨v, n⟩
  else if R≠φ
    call REDUCE
until R=φ and V=φ
call SHIFT
```

REDUCE

```
remove one element ⟨v, n⟩∈R
N⇐LHS(n)
m⇐LENGTH(n)
v'⇐ANCESTOR(v, m)
s⇐GOTO(STATE(v'), N)
if the node v'' already exists
such that STATE(v'')=s and
                    v'=ANCESTOR(v'', 1)
  if the state transition δ(v, ε)=v''
                              already exists
    do nothing
else
  create a new node v''
  such that STATE(v'')=s and
                    v'=ANCESTOR(v'', 1)
  create a new state transition δ(v, ε)=v''
  V⇐V∪{v''}
```

SHIFT

```
for each ⟨v, x, s⟩⊂S
  if the node v' already exists
  such that STATE(v')=s
                and ANCESTOR(v', 1)=v
    if the state transition δ(v, x)=v'
                              already exists
      do nothing
  else
    create a new node v'
    such that STATE(v')=s and
                ANCENTOR(v', 1)=v
    create a new state transition δ(v, x)=v'
```

2.4 Recognition Algorithm

At this point, the formulation of the recognition algorithm is straightforward. The recognition algorithm can be described as follows:

(1) Initialization
- Create an initial state transition network.
- Initialize the likelihood and backpointer buffers.

(2) For each frame, performance of steps (3) through (5)

(3) Dynamic network generation

(4) FSA-based one-pass Viterbi search

(5) Making nodes with low likelihood values inactive

(6) Backtracking along the best sequence of network states

Step 4 performs a conventional FSA-based one-pass Viterbi search to update the likelihoods and backpointers, using a state transition $\hat{\delta}(v, x)$ created by the dynamic network generation algorithm.

3. Experimental Results

We implemented a Japanese speech recognition system based on the recognition algorithm described above, and conducted an experimental performance evaluation.

3.1 Task and Grammar

Evaluation experiments were carried out, using Japanese phrase-wise utterances from the *ATR conference registration task*.[3),18)] This task consists of simulated dialogues between a secretary and participants at an international conference. The CFG for this task has 1,973 production rules, which use 744 Japanese words. In our system, phone models are used as the basic speech unit, and thus phonetic transcriptions for words are included in the grammar. That is, the terminal symbols of the CFG are phone names.

The degree of sophistication of a recognition task is usually measured by the test-set perplexity.[12)] The test-set perplexity is the information-theoretic average branching factor of the language model along the test sentences (test-set). For a CFG, it can be calculated as follows: (1) parse all sentences in the test-set, and count all the distinct words that can follow each word, and (2) compute the geometric mean of word

Table 1 Phrase recognition performance.

Beam width	Accuracy (%)
20	84.1
30	87.0
40	87.0
50	87.8
100	88.1

choices along the test-set. The test-set perplexity of our CFG is 3.57/phone.

3.2 HMMs and Speech Data

In our experimental evaluation, we used simple phone models as the basic unit. They are represented by discrete-type, context-independent HMMs without duration control. We trained a three-loop model for consonants and a one-loop model for vowels, using the *ATR isolated word database*.[18] To represent phone models with less distortion, we used separate vector quantization (multiple codebooks), in which spectrum, LPC cepstral difference, and power are quantized separately.

The speech data used in the evaluation were sampled at 12 kHz, pre-emphasized with a filter having a transform function of $(1-0.97z^{-1})$, and windowed by using a 256-point Hamming window every 9 msec. Next, 12th-order LPC analysis was carried out, and finally the VQ code sequence was generated. For VQ codebook generation, 216 phonetically balanced words were used.

3.3 Results

Table 1 shows the recognition performance for various beam widths. Here, the beam width is equal to the number of active nodes at each frame. As can be seen from the table, the proposed algorithm attained high recognition accuracy even for small beam widths, where the recognition speed was almost real-time. It is sometimes said that context-free grammar constraints are computationally too demanding for use in real-time speech recognition. However, we proved not only that they can be used, but that CFG-based real-time continuous speech recognition can even be attained with a high degree of accuracy.

4. Experimental Comparison with Other Methods

An experimental comparison was made between the LR-parser-driven one-pass search algorithm and the following speech recognition methods:

(1) A one-pass search algorithm using the finite-state approximation for a CFG
(2) The HMM-LR algorithm

4.1 Finite-State Approximation for CFGs

Spoken language systems often use two different grammar formalisms: FSAs for speech recognition and CFGs (or augmented CFGs) for language analysis. To circumvent the problem of maintaing two grammars, a finite-state approximation for a CFG is sometimes used in the recognition stage.[2),17)]

The first method to be compared is the traditional FSA-based one-pass search using the finite-state approximation for a given CFG. In our approximation, an LR parsing table is converted into an FSA by eliminating reduce and goto actions. The actual conversion procedure is as follows:

(1) Construct the canonical collection of item sets using a standard LR parsing table construction algorithm.
(2) If there are two item sets I_j and I_k such that $[X\rightarrow\gamma\cdot]\in I_j$ and $[A\rightarrow\alpha X\cdot\beta]\in I_k$, then make an ε-transition from state j to state k. By this procedure, the CFG is converted into a nondeterministic FSA with ε-transitions.
(3) Finally, eliminate ε-transitions from the FSA created at Step 2, and get a target FSA.

4.2 HMM-LR Algorithm

The HMM-LR speech recognition algorithm[9),10)] is one of the first algorithms to exploit the effectiveness of LR parsing in the speech recognition area. As the name implies, this algorithm tightly couples hidden Markov models and generalized LR parsing, in which the predictive LR parser drives HMMs directly. Two of its major advantages are that it can obtain multiple recognition hypotheses and that explicit HMM duration control is easily incorporated.

The HMM-LR algorithm is similar to the algorithm in this paper in that the LR parser is used for symbol prediction/generation. However, it differs in using the frame-asynchronous tree search to expand partial hypotheses.

The HMM-LR algorithm was adopted for the large-scale speech-to-speech translation project of Advanced Telecommunications Research (ATR), and the implementation of this algorithm is known as the *ATR HMM-LR speech recognition system*.[5),10)] The ATR HMM-LR system introduced many other techniques such

as HMM state duration control and speaker adaptation, and achieved an impressive performance. For fair comparison, we used the core part of this system in the subsequent recognition experiments.

4.3 Experimental Comparison

The three algorithms were compared under the conditions described in Section 3. Because these algorithms use different search strategies and different beam search techniques, it is difficult to compare them directly. We investigated the relationship between the following two factors for various beam widths:

- Average CPU-time needed to recognize one utterance in the test data
- Recognition accuracy

In other words, given the same amount of recognition time, which algorithm will give the most accurate results? Or, equivalently, in attaining the same accuracy, which algorithm is fastest?

From the results shown in **Fig. 6**, we can conclude that the LR-parser-driven one-pass search algorithm outperforms the other two algorithms when the available recognition time is short.

Of course, each algorithm has both strong and weak points, and thus our experiments does not necessarily imply the superiority of our algorithm. Our algorithm produces only the best recognition hypothesis, while the HMM-LR algorithm gives multiple N-best hypotheses. Recently, some methods have been proposed to find the N-best hypotheses in the one-pass search algorithm.[7),19)] We are considering extending our algorithm to the N-best problem.

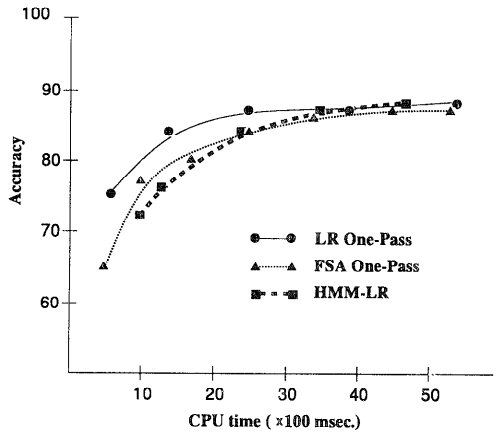


Fig. 6 CPU-time vs. recognition accuracy.

5. Conclusion

This paper has presented a novel continuous speech recognition algorithm that integrates three major technologies: (1) hidden Markov models for speech, (2) a generalized LR parser for handling CFG constraints, and (3) the one-pass search algorithm for achieving an efficient search. The heart of the algorithm is grammar node path-merging based on the LR parser's state stack. To implement path-merging efficiently, we have introduced the shared tree-structured stack. We have also proposed an efficient dynamic network generation algorithm based on the generalized LR parsing technique.

Our evaluation experiments revealed that the proposed algorithm attained high recognition accuracy even for small beam widths, and that CFG-based real-time continuous speech recognition is feasible.

The current implementation uses simple HMMs, namely, discrete-type context-independent HMMs. In future research, we intend to implement a more accurate continuous speech recognition system by incorporating enhanced HMMs with continuous-density observation functions and context-dependent phone models.

REFERENCES

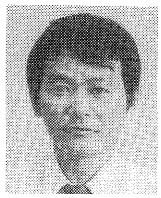
1) Aho, A. V., Sethi, R. and Ullman, J. D.: *Compilers, Principles, Techniques, and Tools*, Addison-Wesley, Massachusetts (1986).
2) Black, A. W.: Finite State Machines from Feature Grammars, *Proc. Int. Workshop on Parsing Technologies*, pp. 277-285 (1989).
3) Ehara, T., Ogura, K. and Morimoto, T.: ATR Dialogue Database, *Proc. Int. Conference on Spoken Language Processing*, pp. 1093-1096 (1990).
4) Godin, C. and Lockwood, P.: DTW Schemes for Continuous Speech Recognition: A Unified View, *Computer Speech and Language*, No. 3, pp. 169-198 (1989).
5) Hanazawa, T., Kita, K., Nakamura, S., Kawabata, T. and Shikano, K.: ATR HMM-LR Continuous Speech Recognition System, *Proc. 1990 Int. Conference on Acoustics, Speech and Signal Processing*, pp. 53-56 (1990).
Also in: *Readings in Speech Recognition*, Waibel, A. and Lee, K.-F. eds., Morgan Kaufmann Publishers, pp. 611-614 (1990).
6) Huang, X. D., Ariki, Y. and Jack, M. A.: *Hidden Markov Models for Speech Recognition*, Edinburgh University Press (1990).

7) 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. 1992 Int. Conference on Acoustics, Speech and Signal Processing*, pp. I-21-I-24 (1992).
8) Kai, A. and Nakagawa, S.: A Frame-Synchronous Continuous Speech Recognition Algorithm Using a Top-Down Parsing of Context-Free Grammar, *Proc. 1992 Int. Conference on Spoken Language Processing*, pp. 257-260 (1992).
9) Kita, K., Kawabata, T. and Saito, H.: HMM Continuous Speech Recognition Using Predictive LR Parsing, *Proc. 1989 Int. Conference on Acoustics, Speech and Signal Processing*, pp. 703-706 (1989).
10) Kita, K.: *A Study on Language Modeling for Speech Recognition*, Ph. D thesis, Waseda University (1992).
11) Lee, C.-H. and Rabiner, L. R.: A Frame-Synchronous Network Search Algorithm for Connected Word Recognition, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. 37, No. 11, pp. 1649-1658 (1980).
12) Lee, K. F.: *Automatic Speech Recognition: The Development of the SPHINX System*, Kluwer Academic Publishers (1989).
13) Monzen, S., Shimizu, T., Singer, H. and Matsunaga, S.: A Study on Frame-Synchronous SSS-LR for Continuous Speech Recognition, *Proc. ASJ Autumn Meeting*, pp. 127-128 (1994).
14) Murveit, H. and Moore, R.: Integrating Natural Language Constraints into HMM-Based Speech Recognition, *Proc. 1990 Int. Conference on Acoustics, Speech and Signal Processing*, pp. 573-576 (1990).
15) Ney, H.: The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. 32, No. 2, pp. 263-271 (1984).
16) Okada, M.: A Unification-Grammar-Directed One-Pass Search Algorithm for Parsing Spoken Language, *Proc. 1991 Int. Conference on Acoustics, Speech and Signal Processing*, pp. 721-724 (1991).
17) Pereira, F. C. N. and Wright, R. N.: Finite-State Approximation of Phrase Structure Grammars, *Proc. 29th Annual Meeting of the Association for Computational Linguistics*, pp. 246-255 (1991).
18) Sagisaka, Y., Takeda, K., Abe, M., Katagiri, S., Umeda, T. and Kuwabara, H.: A Large-Scale Japanese Speech Database, *Proc. Int. Conference on Spoken Language Processing*, pp. 1089-1092 (1990).
19) Soong, F. K. and Huang, E.-F.: A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition, *Proc. 1991 Int. Conference on Acoustics, Speech and Signal Processing*, pp. 705-708 (1991).
20) Tomita, M.: *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems*, Kluwer Academic Publishers (1986).
21) Tomita, M. ed.: *Generalized LR Parsing*, Kluwer Academic Publishers (1991).

(Received July 5, 1994)
(Accepted February 10, 1995)

Kenji Kita received the B.S. degree in mathematics and the Ph.D degree in electrical engineering, both from Waseda University, Tokyo, Japan, in 1981 and 1992, respectively. From 1983 to 1987, he worked for the Oki Electric Industry Co. Ltd., Tokyo, Japan. From 1987 to 1992, he was a researcher at ATR Interpreting Telephony Research Laboratories, Kyoto, Japan. Since 1992, he has been with the Faculty of Engineering, Tokushima University, Tokushima, Japan, where he is currently an Associate Professor. He is a member of the Information Processing Society of Japan, the Acoustical Society of Japan, the Institute of Electronics, Information and Communication Engineers, the Association for Natural Language Processing, and the Association for Computational Linguistics. His current research interests include speech recognition, natural language processing, and corpus linguistics.

Yoneo Yano received the B. E., M.E., and Ph.D degrees in communication engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively. Since 1974 he has been with Faculty of Engineering, Tokushima University, Tokushima, Japan. He is currently a Professor in Information Science and Intelligent Systems. From 1979 to 1980, he was a visiting Research Associate at the Computer-Based Education Research Laboratory, University of Illinois, Urbana, IL., USA. His current interests are in the intelligent CAI, human interface and personal database system. He is a member of the Information Processing Society of Japan, the Institute of Electronics, Information and Communication Engineers, the Japan Society for CAI and IEEE.



Tsuyoshi Morimoto received the B.E. and M.E. degrees in electronics from Kyushu University, Fukuoka, Japan, in 1968 and 1970, respectively. From 1970 to 1987, he worked at NTT Communication Laboratories. In 1987, he joined ATR Interpreting Telephony Research Laboratories, Kyoto, Japan, where he directed the Knowledge and Data Base Department. He is currently the Head of Department 4 in ATR Interpreting Telecommunications Research Laboratories, Kyoto, Japan. He is a member of the Information Processing Society of Japan, the Institute of Electronics, Information and Communication Engineers, the Association for Natural Language Processing, and the Japanese Society for Artificial Intelligence. His current research interests include speech recognition, natural language processing, and the integration of the two.