Proc. ITSC'99

A Generic Framework for Spoken Dialogue Systems and Its Application to a Car Navigation Task

Y. Kono
kono@krl.toshiba.co.jp
Kansai Research Center
Toshiba Corporation
8-6-26 Motoyama-Minami
Higashi-Nada
Kobe 658-0015, Japan

T. Yano
yano@krl.toshiba.co.jp
Kansai Research Center
Toshiba Corporation
8-6-26 Motoyama-Minami
Higashi-Nada
Kobe 658-0015, Japan

M. Sasajima
sasa@krl.toshiba.co.jp
Kansai Research Center
Toshiba Corporation
8-6-26 Motoyama-Minami
Higashi-Nada
Kobe 658-0015, Japan

Abstract

This paper presents EUROPA, a new generic framework for spoken dialogue systems, and its application to a car navigation task. A EUROPA-based system is based on keyword-spotting, i.e., it accepts and understands user's utterance as a set of keyword-sequences, to cope with the diversity of wording in spoken language.

Applying voice interface techniques to practical tasks such as car navigation involves many recognition errors, false alarms in the case of keyword-spotting. EUROPA employs a new parsing algorithm, BTH, which is capable of efficiently parsing a keyword lattice that contains a large number of false alarms. The BTH parser runs without unfolding the given keyword lattice, and thus it can efficiently obtain a set of keyword-sequences acceptable to the given grammar as the parser result.

This paper also presents MINOS, a prototype spoken dialogue system which is built by applying EUROPA to a car navigation task. It understands the user's spoken queries on either places or sections, and replies to them by synthesized voice. It runs on a single Windows-based portable PC in near real time.

1 Introduction

Car navigation with voice interface is desirable for aiding drivers whose eyes, hands, and legs are employed for driving. Recently, some car navigation systems are equipped with not only guidance generation with synthesized voice, but also with operation acceptance with speech recognition. However, voice interface is not popular since it is neither useful nor user-friendly for the typical driver. A voice interface to/with which a user is able to "talk" as if to/with a human is needed. This paper focuses on the following three points.

First, the voice interface should accept spoken language. Application of the voice interface is effective especially when the user cannot use his/her hands because of another task, for example, driving a car, cooking in a kitchen, or operating a power plant. It would be difficult for such users to speak their intention in written language, which involves many grammatical constraints.

Second, it is very important for developing good user interfaces to shorten the duration between users' tests, their feedback to the interface, and next tests. The more tests, the better the interfaces will be.

Lastly, the system with the voice interface should answer users' questions within a short time. So far, most car navigation systems with a voice interface require prompt answers. Slow systems are not useful. We have developed BTH, Bun (meaning "sentence" in Japanese) Template Hash, a parsing algorithm which is capable of efficiently parsing a keyword lattice with a large number of false alarms[4].

Employing the BTH parser, we have developed a new generic framework for spoken dialogue systems, called EUROPA (Environment for building Utterance RecOgnizeable PAckage), which is a package of modules that makes possible easy attachment of a voice interface function to task-dependent systems. A EUROPA-based system accepts and understands the user's utterance which contains non-grammatical terms, unnecessary words, etc, and replies to it in synthesized voice. EUROPA is applied to prototyping of a car navigation system, called MINOS. It answers two types of questions. One is a location of a thing such as a facility, a service-area, a parking lot or a shop. The other is a duration between two locations. All modules are built on one notebook PC(Pentium II 266MHz), which accepts more than 1 million patterns of sentences and in almost all cases answers within 2 seconds.

2 EUROPA

We have designed EUROPA as a generic framework for spoken dialogue systems. Figure 1 shows the overall process of man-machine dislogue in EUROPE. The user's utterance is recognized by the keyword-spotting engine and it generates a keyword lattice as a recognition result. The obtained lattice is parsed by the BTH parser to generate a set of possible keyword-sequences.

0-7803-4975-X/98/$10.00 © 1999 IEEE

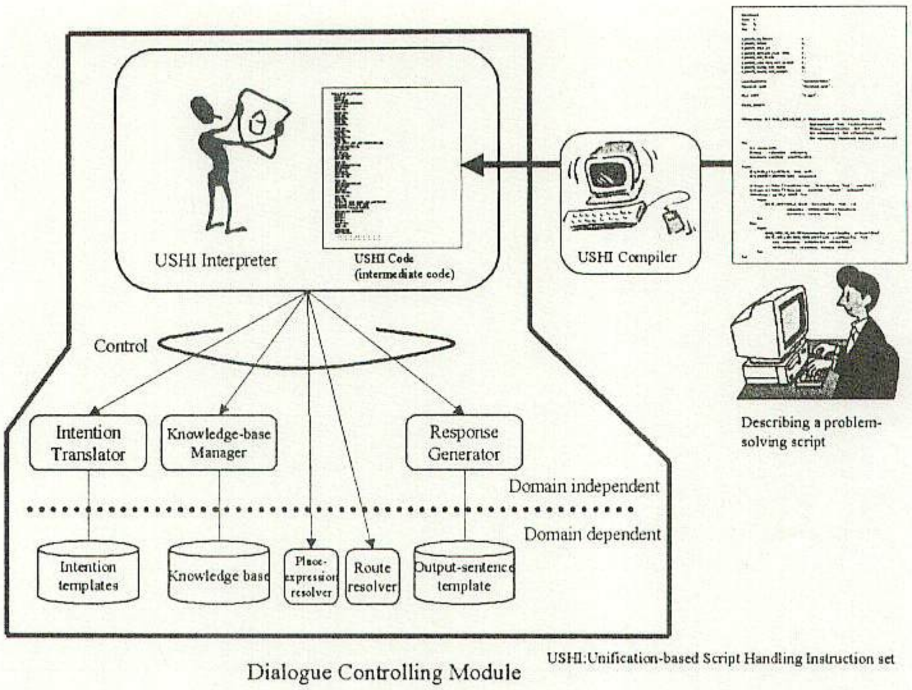

Figure 3: Overall configuration of the dialogue controlling module

destination?"

3 The BTH Parser

We had previously developed TOSBURG-II, a spoken language dialogue system[1, 2]. It picked up the acceptable keyword-sequence as a sentence from the keyword-lattice obtained from a keyword-spotting engine for the task of selling hamburgers with a very limited vocabulary, and performed the semantic analysis / intention understanding. The system recognized only meaningful words, i.e., nouns and verbs, received a plausible keyword-sequence as a sentence, and responded to the user by synthesized voice. The TOSBURG-II study revealed the following: (1) by applying keyword-spotting technology, it is possible to cope with phenomena characteristic of spoken language such as unnecessary words or unfixed forms, (2) false alarms which are unavoidable in the keyword-spotting process can be also corrected by sentence analysis.

In order to expand the scale of the task which a spoken language dialogue system is capable of executing and to apply the system for practical tasks, it is necessary to correspond to a large vocabulary and a large-scale grammar. However, expansion of vocabulary and grammar causes the number of recognized words and the size of the corresponding keyword lattice to increase explosively. This poses a problem in that the processing time required for analysis becomes huge.

We have developed BTH, Bun (meaning "sentence" in Japanese) Template Hash, a parsing algorithm which is capable of efficiently parsing a keyword-lattice with a large number of false alarms[4]. When a keyword lattice is large, the list of possible keyword-sequences generated by unfolding the lattice is huge too. Parsing the candidates in the list one by one causes an explosion of calculation time. The BTH parser runs without unfolding the given keyword lattice. This parser (1) holds hash tables for each sentence-type template, (2) calculates the set of sentence-types to which each node in the lattice can belong by referring to the table, and (3) propagates the list to the nodes following each node. It can efficiently obtain a set of keyword-sequences acceptable to the given grammar as the parser result.

We have implemented this algorithm on PCs and tested the method by applying it to a car navigation task. In this work, the vocabulary for recognition exceeds 700 words, and the keyword lattice that can generate over 1,000,000 candidate keyword-sequences is usually obtained as a recognition result. Also, parsing of such a large-scale lattice can usually be completed in real time.

Our research goal is to develop a generic framework for spoken interfaces that runs on practical applications with considerable scale of task. We cannot expect a 100% recognition rate with current speech recognition technologies, even if strong constraints are applied as a language model. It is especially difficult to correctly recognize bound words, i.e., words that have no meanings themselves, e.g., postpositions. We have designed our speech understanding mechanism based on the following policies:

- Embodying scalability so that it works well in practical applications.
- Analyzing/understanding a user's spontaneous utterance with the grammar that accepts a sentence as a keyword-sequence consisting only of free words, i.e., words that have meanings themselves, e.g., nouns and verbs.
- Recognizing a user's utterance by the keyword-spotting technique. A list of possible keyword-sequences is obtained by parsing a keyword lattice generated from a keyword-spotting result.

To realize our goal, it is crucial to establish the technology for post-recognition process that can efficiently extract plausible sentences/keyword-sequences by dealing with the keyword lattice obtained from the keyword-spotting engine. We have developed the BTH parser which is capable of efficiently parsing a keyword lattice that contains a large number of false alarms[4]. This section describes its features and techniques.

3.1 Keyword-Lattice Parsing Problem

One technique for parsing a lattice structure that is obtained by speech/handwriting recognition is to sequentially apply a natural language parser to candidate sentences one by one which are generated by unfolding the lattice. The TOSBURG-II employed a generalized LR parser[3], which analyzes a keyword lattice generating midway candidates and pruning hopeless paths[1, 2]. The keyword lattice, however, generated by a speech recognizer with a large volume of vocabulary and grammar, contains a complicated structure with many nodes and forms in general. It is hard to analyze each candidate generated by unfolding the lattice in the light of both calculation time and memory.

It is essential to retrieve a huge amount of both vocabulary and grammar to develop a speech interface for a practical application. We collected 389 example sentence utterances that are natural and needed by drivers who use car navigation systems with speech recognition. They include actual utterances made by one of us while driving a car and while being a passenger in a car. Then we analyzed them and made grammatical rules that apply to those examples. We also designed structures of the user's intentions by classifying the collected utterances into 5W (Where, What, Which, When, Why) and 2H (How, How-Much).

In parallel with the analysis of sentence patterns and grammar, we picked up keywords from the example utterances. For instance, the sentence "Deguchi-no-mae-no-saigo-no-service-area-ha-doko? (Where is the last service area before the exit?)" was classified into the sentence-type group of "Where," and five words, i.e., deguchi (exit), mae (before), saigo (the last), service-area, and doko (where), were selected as keywords. As a result of the analysis, about 700 keywords were picked up as recognition vocabulary and were classified into about 110 word-classes (parts of speech).

As the vocabulary for speech recognition becomes larger, the number of false alarms in a recognition result increases and the corresponding keyword lattice becomes huge. Figure 4 depicts the example keyword lattice which is obtained as the speech recognition result when a user uttered the sentence above ("Deguchi-no-mae-no-saigo-no-service-area-ha-doko?"). 36 keywords, i.e., 5 correct and 31 false alarms, were spotted in this example. When only connectable time constraints are applied, over 4 million possible keyword-sequences can be generated, each of which is a different path from the start node to the end node, if the lattice is unfolded. Thus a lattice parser, which is capable of parsing without unfolding a given keyword lattice and generating a set of acceptable keyword-sequence candidates by given grammar, is required.

3.2 Parsing a Keyword-Lattice with Template Hash Tables

Based on the following viewpoints, we have developed the BTH (Bun [meaning "sentence" in Japanese] Template Hash) parser, which is an efficient lattice parser for keyword-spotting that satisfies the above-mentioned requirements:

- *Real-time processing in the scale of practical applications using conventional calculation power:* Both more vocabulary and less calculation are required to develop a practical speech interface. We aimed to realize a parser working in real-time on 700-800 MIPS RISC MPUs, i.e., the MPUs that are expected to be applied in next-generation car navigation systems.
- *Grammatical representation taking into account both the recognition method and the application:* It is difficult to construct a comprehensive and accurate grammar for actual utterance even if the task to which it is applied is limited, because word order and syntax in actual utterance are wide-ranging. A weakly constrained grammatical representation is required to cope with noticeable phenomena in speech, such as inversions. On the other hand, it is pointless for the parser to accept sentences which the application program cannot execute. A grammatical representation that generates very complicated sentences, e.g., one employing recursive rules, would be too expressive for practical use.
- *Parsing without unfolding the given lattice:* In the case of a large vocabulary, a recognition result lattice from a keyword-spotting engine includes many false alarms as mentioned above. It costs huge calculation time if the parser analyzes unfolded candidates one by one. Therefore, a lattice parser, which is capable of parsing without unfolding a given keyword lattice, is required.

The BTH parser embodies the following features:

- *Grammatical representation based on templates of word-class sequences:* The grammar is represented as a set of templates, each of which forms

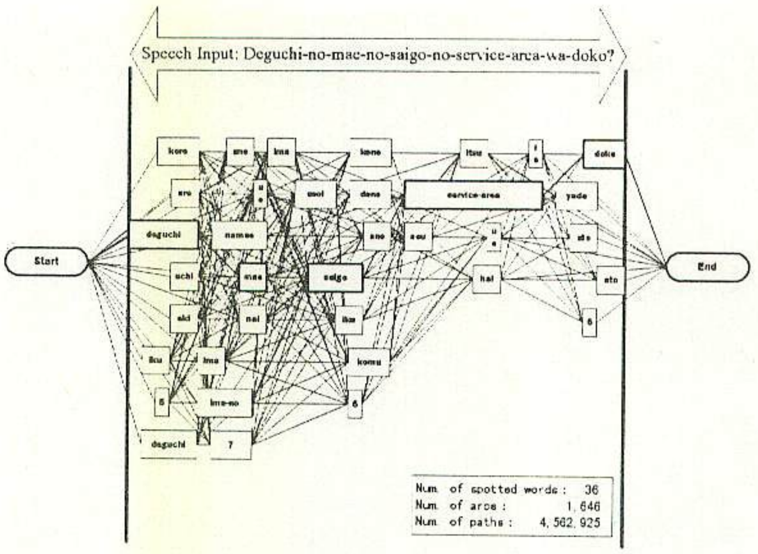

Figure 4: An example of a keyword lattice

a word-class sequence. An interface designer collects sentences in a task and classifies them into sentence-types, i.e., word-class sequences. The parser analyzes a keyword lattice utilizing the template sets. The grammar must be represented non-recursively.

- *Generation of a hash table from the sentence-type template:* As a pre-process of the parsing, the given sentence-type template is compiled into a hash table, whose element is a correspondence between the appearance of a certain word-class in sentences and the established set of sentence-types which applies to the word-class. A Bun (meaning "sentence" in Japanese) Template Hash table is composed of representations, each of which means "if a word of a certain word-class is recognized at a certain order in a sentence, the word of the order applies to either of certain sentence-types."

- *Parsing of a lattice by propagating belonging sentence-type information:* The parser scans the given lattice from the start node. At each node, the set of sentence-types, to which a certain node in the lattice belongs, is calculated by examining set calculations between the corresponding elements in the hash table and the set of sentence-types propagated from the nodes connected just in front of the node. Then the set is propagated from the node to connecting node(s). By repeating the method, the set of sentence-types to which the given lattice belongs is finally obtained.

The detailed parsing process is described in the next subsection.

3.3 Parsing Process

To parse a keyword lattice, the BTH parser utilizes four kinds of information structure, i.e., a dictionary of word-classes and words, a dictionary of patterns of word-class sequences, a sentence-type hash dictionary, and a list of processing nodes. The list of processing nodes is the set of nodes that are the current targets of processing. Each node in the lattice holds three kinds of data, i.e., a list of antecedent nodes, a list of unprocessed antecedent nodes, and an inter-processing list, composed of sets of sentence-types, each of which is the sentence-type to which the node belongs at a certain order.

Given a new lattice to analyze, at the beginning of the parsing process, the BTH parser (1) copies the list of antecedent nodes to the list of unprocessed antecedent nodes for each node in the lattice, (2) adds all the nodes which may possibly be heads of result keyword-sequences to the list of processing nodes, and (3) sets the initial score that is obtained from the sentence-type hash dictionary for each inter-processing list of the node in the list of processing nodes.

Then the parser repeats the propagation process while the list of processing nodes is not empty. In the propagation process, the following steps are applied to each node in the list of processing nodes:

1. Take one node as the current target from the list of processing nodes and check whether the list of unprocessed antecedent nodes is empty. If it is not empty, check the next node in the list of processing nodes.

2. If the list of unprocessed antecedent nodes of a node X in the list of processing nodes is empty, propagate the inter-processing list of X to all suc-

ceeding nodes and add them to the list of processing nodes, and then remove X from the list.

3. At each node to which the inter-processing list of its antecedent node is propagated, re-calculate the inter-processing list.

If the list of processing nodes is empty when step 3 above is completed, the inter-processing list of the end node is the set of acceptable sentence-types by the lattice. The parser result, i.e., the set of acceptable keyword-sequences by the lattice, is obtained from the set of sentence-types by re-scanning the lattice.

The cost of the entire parsing process can be estimated from the cost of set calculations to obtain the inter-processing list for each node. The maximum possible number of set calculations, i.e., unions and intersections, is equal to the number of links in the lattice times the number of words in the longest keyword-sequence. It is the worst case, however. Much less calculation is required in most cases.

4 MINOS

We have implemented a spoken dialogue system based on EUROPA, called MINOS, which is able to respond to queries respecting locations made by a user while driving a car. Modules in MINOS run collaboratively, communicating with each other via socket-based messaging. As shown in Figure 5, the application module of MINOS has a map-based appearance, which is based on ProAtlas, a map software developed by ALPS Mapping Co., Ltd. It simulates car driving by re-playing pre-recorded position data obtained from the Global Positioning Satellite (GPS) system.

The application module simulates driving a car and continuously notifies the dialogue module of the simulated current position at regular intervals of time. When a user asks a question in Japanese, e.g., "Deguchi-no-mae-no-saigo-no-service-area-ha-doko? (Where is the last service area before the exit?)", the keyword-spotting engine recognizes the utterance and notifies the dialogue controlling module of a word lattice as depicted in Figure 4.

MINOS analyzes the word lattice by employing the BTH parser and obtains a set of candidate keyword-sequences, which is sorted in the descending order of the initial score of each candidate, which is calculated from phonetic scores of words of the candidate. Each keyword-sequence candidate is converted into a user's intention in the form of a typed feature structure, and is resolved by using current position information and the knowledge base. The knowledge base is a semantic network that contains all the knowledge required to solve questions about locations on the displayed map.

The score of each candidate is revised taking account of the cost of problem solving, and the list of the candidates is reordered. As a result, the candidate with the best score is selected and the text of the answer corresponding to the question is generated, e.g., "Amagasaki Service Area is the last service area before the Nishinomiya Interchange," and the dialogue controlling module notifies the synthesizer of it.

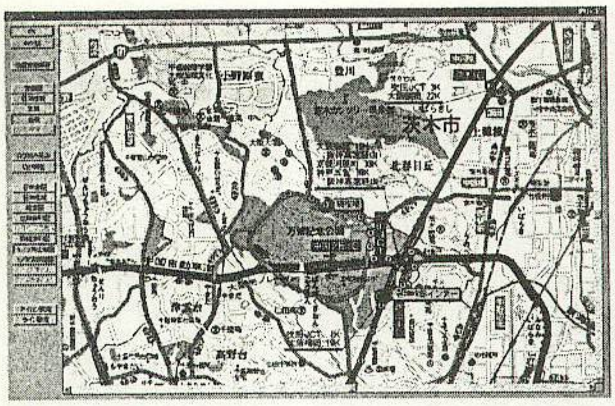

Figure 5: Sample screen of MINOS

5 Concluding Remarks

We have designed a generic framework for spoken dialogue systems, EUROPA, which employs an efficient keyword lattice parser, BTH, and developed a PC-based spontaneous speech interface for a car navigation task, MINOS. MINOS processes over 700 words of recognition vocabulary. MINOS is able to respond to a user's question within a few seconds of the question being asked on a portable PC.

In the case of the task, it is common for over 100 spotted words to be notified from the recognition engine, and consequently over 1 million possible keyword-sequences can be generated by unfolding the corresponding lattice even if word-class bigram is applied to the lattice. Our method, however, is able to parse such a large lattice in a practical time. The result shows promise respecting implementation of the function of spontaneous speech interface in next-generation car navigation systems equipped with RISC MPUs of about 700MIPS.

References

[1] Takebayashi, Y., Tsuboi, H., Kanazawa, H., Sadamoto, Y., Hashimoto, H., and Shinchi, H., "A Real-Time Speech Dialogue System Using Spontaneous Speech Understanding." *IEICE Trans. Inf. & Syst.*, E76-D: 112-120, 1993.

[2] Takebayashi, Y., Tsuboi, H., and Kanazawa, H., "Keyword-Spotting in Noisy Continuous Speech Using Word Pattern Vector Subabstraction and Noise Immunity Learning." *Proc. ICASSP '92*, II-85-88, 1992.

[3] Tomita, M., "An Efficient Word Lattice Parsing Algorithm for Continuous Speech Recognition." *Proc. ICASSP '86*, 1569-1572, 1986.

[4] Kono, Y., Yano, T., and Sasajima, M., "BTH: An Efficient Parsing Algorithm for Word-Spotting." *Proc. ICSLP '98*, 2067-2070, 1998.

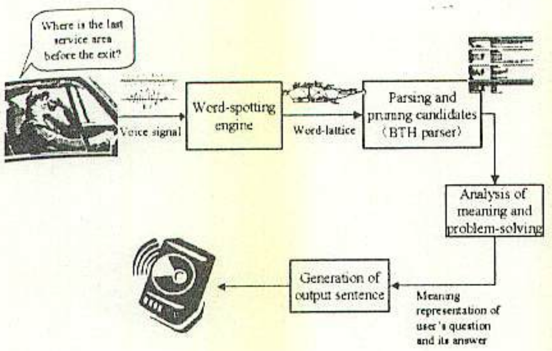
Where is the last service area before the exit?
Voice signal
Word-spotting engine
Word-lattice
Parsing and pruning candidates (BTH parser)
Analysis of meaning and problem-solving
Meaning representation of user's question and its answer
Generation of output sentence

Figure 1: Dialogue process

each of which is acceptable by the given grammar. The set is analyzed and solved in terms of meaning and the most plausible keyword-sequence in the set is selected. Finally, the reply to the user's question is played back by synthesized voice.

To accept spoken language, our voice recognition module does keyword-spotting. The sequence of spotted keywords represents the user's intention. In the case of spoken Japanese, misuse or loss of particles often occurs. Keyword spotting does not deal with them, which simplifies acceptable grammar rules. This characteristic also solves the problems of dealing with unnecessary terms such as "aah" or "well." Just by excluding them from the keyword set, we can accept sentences with these words. Further more, change of word order which also occurs in Japanese spoken dialogue can be dealt with easily. The BTH parser is employed for efficiently parsing the keyword lattice which is obtained by keyword-spotting, and it is transformed into a set of possible keyword-sequences. The detail of the BTH parser is described in Section 3.

Obtaining the set of possible keyword-sequences from the BTH parser, the dialogue controlling module processes it to generate a reply. The overall configuration of the module is depicted in Figure 3. Firstly, it transforms the given set into a set of representations of input intention, each of which corresponds to a keyword-sequence candidate, by employing Intention Translator. Then it resolves a user's question by referring to the knowledge base and generates meaning representation of the query which corresponds to the most plausible keyword-sequence and its answer. Obtaining the information, Response Generator generates responding sentences. The problem-solving process is based on unification of feature structures.

To shorten prototyping and testing cycle EUROPA separates modules into two groups. One consists of the domain dependent modules such as a word dictionary, grammar rules, rules for translating user intentions, rules for generating sentences as an answer to the user input, and domain-specific problem solvers, such as Place-expression Resolver and Route Resolver. The

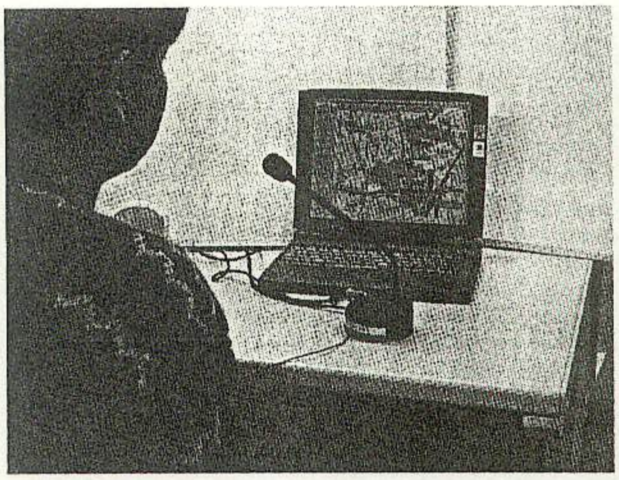

Figure 2: Overview of MINOS

other consists of domain-independent modules such as a lattice parser, an interpreter for translating user intentions, and an interpreter for generating answer sentences.

To run a dialogue, the dialogue system must control a set of modules which belong to one of the two groups. For example, a car navigation task which solves a location specified by a user utterance, requires not only a generic parser but also a domain-specific problem solver which resolves the location the user intended. Embedding such control codes including domain-specific parts reduces portability of the framework. EUROPA solves this problem by adopting an interpreter for such codes that manage dialogue. We call this interpreter "USHI" which stands for "Unification-based Script Handling Interpreter." A system developer describes the process of meaning analysis, problem-solving, and response generation in an USHI script, and the module interprets the script. To gain enough speed for the response, the USHI script for managing problem solving is compiled beforehand into another form to be interpreted faster as shown in Figure 3.

EUROPA is applied to prototyping a car navigation system, called MINOS. Figure 2 shows the overview of it. A sample screencopy of MINOS is also shown in Figure 5. The system answers two types of questions. One is a location of a thing such as a facility, a service-area, a parking lot or a shop. The other is a duration between two locations. All modules are built on one notebook PC(Pentium II 266MHz), which accepts more than 1 million patterns of sentences and in almost all cases answers within 2 seconds. In selecting sentences to answer from input lattice, the system refers to their scores that are calculated in terms of phonetic value and meaning. One more function of EUROPA is asking back to the user when the score of the sentence is lower than a given threshold such as "Did you ask about a restaurant just before the