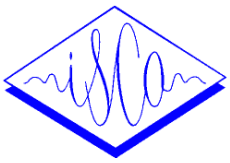

6th International Conference on Spoken Language Processing (ICSLP 2000)
Beijing, China
October 16-20, 2000

Goal-oriented Table-driven Design for Dialogue Manager

Huei-Ming Wang and Yi-Chung Lin

Advanced Technology Center,
Computer and Communications Laboratories,
Industrial Technology Research Institute,
Chutung, Taiwan 310,
{hmw, lyc}@itri.org.tw

ABSTRACT

Spoken dialogue systems have strong potentiality in becoming the main stream of next generation man/machine interface. To meet this future, spoken dialogue systems must not only be portable to support various applications, but also be scalable to provide capable dialogue to handle sophisticated services. In this paper, we proposed a goal-oriented table-driven approach to design a portable and scalable dialogue manager. In this approach, the dialogue strategy is modeled by a set of tables. Each table in the set handles a simple sub-goal and the tables are organized hierarchically according to a predefined flow to achieve a particular goal. The data sharing property of the proposed approach makes tables become reusable and, consequently, greatly relieves the efforts of scaling-up an old application or developing a new one. In addition, the mix-initiative dialogue management is pursued by incorporating both the system-initiated and the user-initiated table switching mechanisms.

1. INTRODUCTION

In recent years, because of the mature of speech recognition technology, man-machine interactions have shifted rapidly from simple touch tone menus to more complex speech-based systems. By making use of the communication power of natural language, spoken dialogue systems are developed in several useful domains, such as the timetable inquiry system [1] or the automobile information system [2]. Spoken dialogue systems have strong potentiality in becoming the main stream of next generation man/machine interface. To meet this future, spoken dialogue systems must not only be portable to support various applications, but also be scalable to provide capable dialogue to handle sophisticated services. Among the typical processing modules of a spoken dialogue system, such as speech recognition, language understanding, dialogue management, language generation and speech synthesis, the dialogue management module is the most domain-dependent one. Therefore, it is difficult to design a dialogue manager for supporting a portable and scalable dialogue system.

In the past, several approaches have been proposed to manage man/machine interactions for dialogue system. One of the well-known approaches is the finite state network (FSN) [3]. In this paradigm, each node in the network represents a particular dialogue state, and the links between two nodes indicate possible transitions between dialogue states. The dialogue manager controls interactions by traversing the network from initial state to final state. The merit is that FSN is a formal model like a dialogue grammar and the dialogue flow is straightforward to design. However, as the perplexity of the task or the flexibility of the dialogue increasing, the number of states and links will increase dramatically; consequently, revising and maintaining the network becomes rather difficult. Stochastic machine learning technique is introduced to learn the network automatically [4], however, it's hard to cover new dialogue incrementally because the learning process requires a huge mount of training data.

Another popular approach is form-based approach [2] for its facilities in planning dialogue. The form contains slots of input semantic frame and the relevant constraints of the domain. According to the status of the form, the dialogue manager guides the user to fill the form with prompt associated to each slot. This approach avoids enumerating all possible dialogue states to model the dialogue and allows the user not to comply with the system's prompt. Although the approach is able to build simple information access systems rapidly, it's difficult to be adapted for domains requiring more sophisticated dialogue management [5]. Extended from form-based approach, table-driven approach provides a mechanism to model dialogue by a set of rules with trigger conditions and reactions [6,7]. The trigger condition, the same as status of form in form-based approach, is assigned by the constraint on each slot in grids of table. Table-driven approach inherits all the advantages of form-based approach, in addition, it can specify interested dialogue status in a clearer format and deal with more complicated dialogue. However, considering an application of multi-tasks, the table would become too large to understand and maintain for keeping all the knowledge about handling the conversations that guide users to complete the complicated application.

Although the table-driven approach makes much progress on portability and maintainability, it still has a potential problem of scalability. To provide sophisticated dialogue management for a complex dialogue system, the table that drives the dialogue manager needs to incorporate many parameters and tends to be too large to handle. To resolve this problem, we propose a goal-oriented design to enhance the scalability of the table-driven approach. The basic idea of the design is using the divide-and-conquer tactic. In general, the dialogue process between man and machine is a goal-satisfaction process. The user interacts with the machine with a goal in mind and the machine tries to satisfy it. During the interaction, it is natural for both sides to decompose a complex goal into several simple sub-goals so that each of them can be more easily satisfied by sub-dialogue. In the proposed approach, the dialogue strategy is modeled by a set of tables. Each table in the set handles a simple sub-goal and the tables are organized hierarchically according to a predefined flow to achieve a particular goal. In this approach, the size of each table is moderate and the goal-oriented tables can build up

a dialogue strategy to handle a complicated task in a more clear way.

In addition, to allow flexible man-machine interactions, the mixed-initiative dialogue management strategy is adopted in our approach. Users are not obliged to follow predefined flow to complete a particular goal. They can change their intentions during interactions. Therefore, a user-initiative table switching mechanism is incorporated to capture the new intention and interrupt the ongoing dialogue flow. This mechanism will store the status of the interrupted dialogue flow in a stack so that the interrupted flow can be continued later. Then, according to the new intention and the trace of the ongoing flow, a suitable sub-goal table will be selected to handle new intention. Once new intention is satisfied, the system will switch back to the interrupted flow by popping up the topmost status from the stack.

This paper is organized as follows. First, the table-driven approach is overviewed in Section 2. Then, the concept of goal-oriented design is introduced in Section 3. Section 4 presents the details of the portable dialogue controller which supports the proposed goal-oriented table-driven framework. Finally, the conclusions are made in Section 5.

2. TABLE-DRIVEN APPROACH

In our past research, we proposed a domain-transparent framework to meet the portability and maintainability requirements of dialogue management. In this framework, domain-dependent factors that affect the system responses are identified and encapsulated into an external table. The dialogue control is simplified as a domain-independent, thus portable, table look-up procedure. In addition, by representing the knowledge of how to control dialogue flow in a table form, this design make system developers more easily design and revise the dialogue strategies than the conventional finite-state-network approach.

In our design, the external table, named *task description table* (TDT), is composed of a set of dialogue states, each of them being associated with an action and a trigger condition. The trigger condition of a dialogue state is specified in terms of the domain-dependent factors, named *parameters*. For example, in the weather forecast information system, the domain-dependent factors would be "date", "location", and "topic" (e.g., temperature, relative humidity, etc.). Figure 1 shows a simplified TDT for a weather forecast information system. Three parameters, *location*, *date* and *topic*, are used and six dialogue states are specified with different conditions. The possible conditions include valid parameter (denoted by "+"), void parameter (denoted by "–"), parameter greater than, less than or equal to a value (denoted by ">", "<" and "=", respectively). The parameter is "don't care" if no condition is specified. The actions of states are listed on the right hand side of the table, including one welcome action, three actions for asking more information, one query and a warning action. On the left side, the marker "P" denotes S1 and S2 have higher priority and the marker "G1" denotes S2, S3 and S4 are grouped together. For detailed discussion on using priority and group markers, please refer [7]

		Loc.	Date	Topic	
P	S1	–	–	–	Welcome()
G1	S2	–	×	×	Ask(Loc.)
G1	S3	×	–	×	Ask(Date)
G1	S4	×	×	–	Ask(Topic)
	S5	+	+	+	Query()
P	S6	×	> 7	×	Warn(Too_long)

Figure 1: TDT of a weather forecast information system.

This table-driven approach, despite portable and maintainable, has a potential problem of scalability. To provide sophisticated dialogue management for a complex dialogue system, the TDT needs to incorporate many parameters and tends to be too large to handle. To resolve this problem, we propose a goal-oriented design to enhance the scalability of the table-driven approach.

3. GOAL-ORIENTED DESIGN

The basic idea of the proposed goal-oriented design is using he divide-and-conquer tactic. In general, the dialogue process between man and machine is a goal-satisfaction process. The user interacts with the machine with a goal in mind and the machine tries to satisfy it. During the interaction, it is natural for both sides to decompose a complex goal into several simple sub-goals so that each of them can be more easily satisfied by sub-dialogue. For example, to satisfy the goal of booking train tickets, the system has to get the user's itinerary (sub-goal) about the departure and destination cities, date and time first. Then, depending on the itinerary, the proper scheduled train run will be found by checking train timetable. Moreover, the seat needs to be arranged (sub-goal) according to the user's preference about window-seat or aisle-seat. After getting sufficient information to book the ticket, the system has to

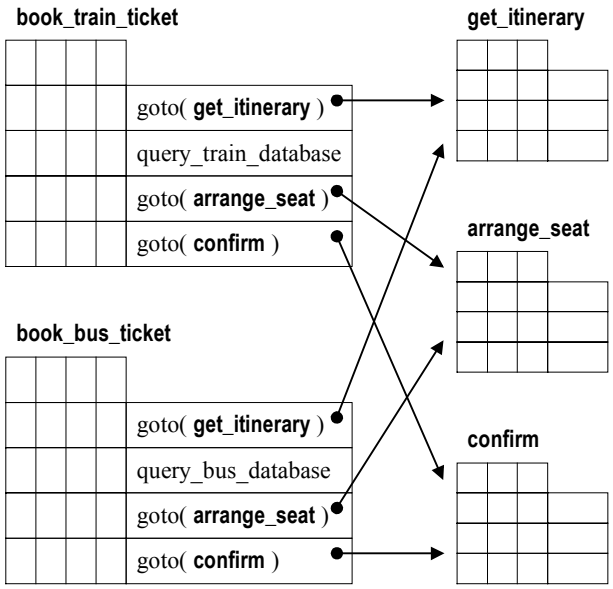

Figure 2: A simplified example of hierarchical task description table set (HTDTS).

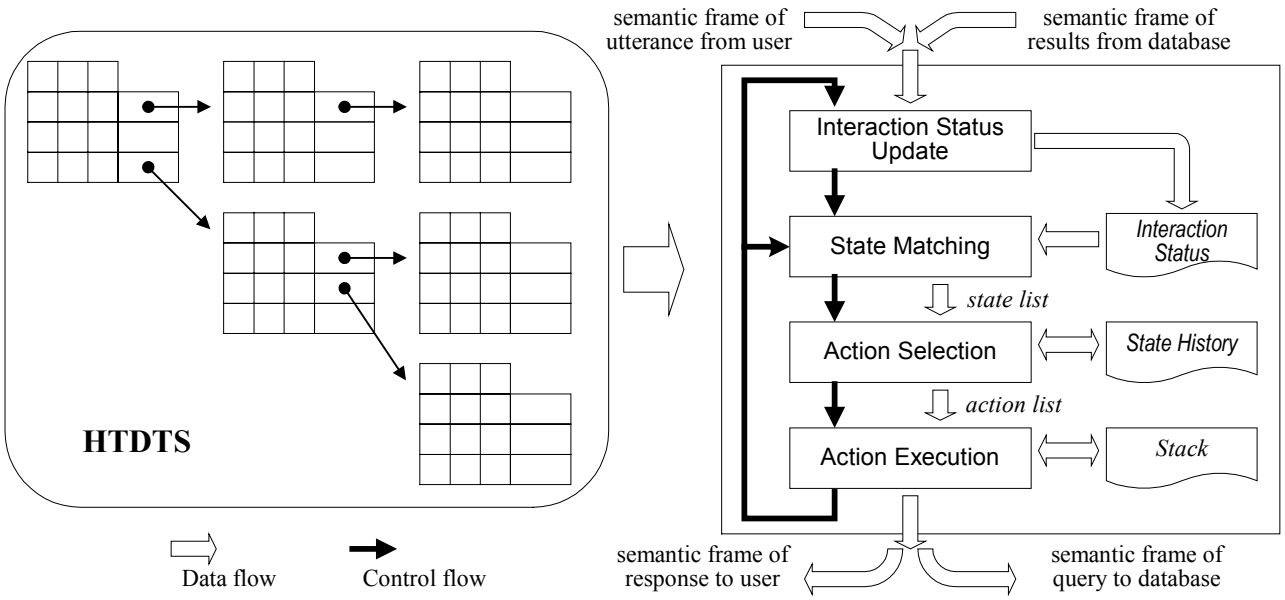

Figure 3: Block diagram of dialogue management controller.

confirm (sub-goal) with the user whether he/she booked the right ticket.

Since the table-driven approach is good at handing simple single-goal task, we propose to manage the dialogue flow with a set of TDTs. Each TDT in the set handles a simple goal and the TDTs are organized hierarchically according to a predefined flow to achieve some complex goals. The set of TDTs is named *hierarchical task description table set* (HTDTS). Figure 2 is a simplified example of HTDTS. In this example, the TDT of the goal "book_train_ticket" is linked to the TDTs of the sub-goals "get_itinerary", "arrange_seat" and "confirm". The action of traveling from a TDT to another likes the behavior of opening a sub-dialogue to satisfy a sub-goal. One should notice that the order of the links in figure 2 does not imply the order to satisfy the corresponding sub-goals. When to traveling to a child TDT depends on the trigger condition of the dialogue state. The next section will give more detailed discussion about how to execute the actions in a TDT.

Furthermore, Figure 2 also clearly shows that TDTs can be shared under the HTDTS paradigm. This is due to the fact that some sub-goals of different goals are the same. This data sharing property makes TDTs become reusable objects and, consequently, greatly relieve the efforts of scaling-up an old application or developing a new one.

4. DOMAIN-INDEPENDENT CONTROLLER

In our design, the dialogue controller is responsible for monitoring the semantic frame of the user's utterance and the semantic frame of the results of database query and, then, consulting the information of the HTDTS to find an appropriate response for the input semantic frame. The response may be a query to the database or an utterance to the user. Since all domain-dependent factors for arranging dialogue flow are encapsulated in the HTDTS, the control kernel of dialogue management can be designed as domain-independent and, thus, portable. Similar to the operation of the controller driven by a single table, the basic operation of the dialogue controller driven by HTDTS is to find the most appropriate action by matching current interaction status with the dialogue states in the HTDTS. To achieve this end, the controller is decomposed into four modules, as shown in Figure 3.

At the very beginning, the controller dynamically constructs its internal data members according to the HTDTS. For example, it constructs its interaction status data member as an array of all parameters in HTDTS. Then, after it receives the first input semantic frame, it activates the root table in the HTDTS and enters the *Interaction Status Update* module to begin the control loop. This module is responsible for updating the *interaction status* according to the semantic frame of user's utterance or the results from querying the database. For example, if the user says "tell me the weather tomorrow", the utterance will be analyzed to a semantic frame whose "Date" slot has the value "tomorrow" and "Topic" slot has the value "weather". As the dialogue manager receive the semantic frame, the interaction status will be updated as follows.

Parameter	Date	Loc.	Topic
Value	tomorrow	–	weather

After updating the interaction status, the *State Matching* module is triggered to find a set of dialogue states by comparing the interaction status with the dialogue states in the active table. The matched dialogue states are sent to the *Action Selection* module to decide which dialogue states should be activated according to the priority and group attributes of dialogue states. Afterward, the actions of active dialogue states are collected and pass to *Action Execution* module to be executed.

4.1. System-initiated Table Switching

In our approach, every table in the HTDTS aims at satisfying a particular goal. If the goal is too complicated, it will be decomposed into a set of sub-goals. That is, a TDT can link to a

set of other TDTs, each of which tries to satisfy a sub-goal of its parent TDT. The link between two TDTs is realized by the special action, *goto*(*x*), where *x* is the name of a TDT. This action simulates the behavior of opening a sub-dialogue to satisfy a sub-goal.

Once a *goto*(*x*) action is executed, the controller will push current TDT (i.e., the active one) into the stack of pending tables. Then, after inactivating current TDT and activating the child TDT, the controller will enter the *State Matching* module. In contrast to the *goto*(*x*) action, a special action, named *return*(), is designed to switch back from the child TDT to its parent. When the *return*() is executed, the controller will inactivate current table and pop out the topmost pending TDT from the stack and activate it.

4.2. User-initiated Table Switching

Through the design of HTDTS, the dialogue manager can complete the user's goal by satisfying some sub-goals in a predefined order. For example, with adequate condition setting of dialogue state, the dialogue manager may plan to book tickets for users by the sequence of asking the itinerary, arranging seats, querying database and making confirmation. However, to allow flexible man-machine interactions, users are not obliged to follow predefined flow to complete a particular goal. They can change their intentions during interactions. For instance, the user could intend to initiate a sub-dialogue for arranging seats while the system with a sub-dialog of getting itinerary information opening. To achieve this end, a user-initiated table switching mechanism is proposed to capture the new intention and interrupt the ongoing dialogue flow.

In our approach, a special parameter, named *user_itention*, is used to monitor the user's current intention. This parameter is set by the language understanding component according to the user's utterance. Once, the user's intention does not equal to the value of *user_itention* parameter in dialogue status, the user-initiated table switching process is triggered. Like the system-initiated table switching process, the controller will push current TDT into the stack of pending TDT and, then, inactivate current TDT. Afterward, by tracing back the TDTs in the TDT-pending stack, the controller will search a new TDT which contains dialogue states match the user's intention. Finally, the controller will activate the new TDT and enter the *State Matching* module.

5. CONCLUSIONS

In this paper, we proposed a goal-oriented design to enhance the scalability of the table-driven approach. In our design, the dialogue strategy is modeled by a set of task description tables (TDTs). Each table in the set handles a simple sub-goal and the tables are organized hierarchically according to a predefined flow to achieve a particular goal. The data sharing property of the proposed approach makes TDTs become reusable and, consequently, greatly relieves the efforts of scaling-up an old application or developing a new one. In addition, the mix-initiative dialogue management is pursued by incorporating both the system-initiated table switching mechanism and the user-initiated table switching mechanism.

This approach has been used to improve our dialogue manager which managed a weather information dialogue system by using a single task description table. Currently, the dialogue manager is driven by a HTDTS which contains 7 TDT and can handle more sophisticated interactions. The successful experience of improving our dialogue makes us believe that the proposed goal-oriented table-driven approach has potential to build up large dialogue system.

6. ACKNOWLEDGEMENT

This paper is a partial result of Project 3P11200 conducted by ITRI under sponsorship of the Ministry of Economic Affairs, R.O.C.

7. REFERENCES

1. Aust, H., Oerder, M., Seide, F. and Steinbiss, V., "The Philips Automatic Train Timetable Information System," *Speech Communication 17, 1995*, pp. 249-262.
2. Goddeau, D., Meng, H., Polifroni, J., Seneff, S., and Busayapongchai, S., "A Form-Based Dialogue Manager for Spoken Language Applications," *Proceedings of ICSLP'96*, pp. 701-704, Philadelphia, 1996.
3. Kita, K. et al., "Automatic Acquisition of Probabilistic Dialogue Models," *Proceedings of ICSLP'96*, pp. 196-199, Philadelphia, 1996.
4. Levin E. et al., "Using Markov Decision Process for Learning Dialogue Strategies," *Proceedings of ICASSP'98*, pp. 201-203, Seattle, 1998.
5. Constantinides, P.C., Hansma, S., Tchou, C., Rudnicky, A.I., "A Schema Based Approach to Dialog Control," *Proceedings of ICSLP'98*, pp. 409-412, Australia, 1998.
6. Seneff S., "Discourse and Dialogue Modeling in the Galaxy System," *Proceedings of Dialogue System and Discourse Analysis Workshop*, pp. 12-24, Taipei, Taiwan, 1997.
7. Lin, Y.-C. et al., "The Design of a Multi-Domain Mandarin Chinese Spoken Dialogue System," *Proceedings of ICSLP'98*, pp. 41-44, Australia, 1998.