DIALOGUE MANAGEMENT
BASED ON A HIERARCHICAL TASK STRUCTURE

Jiang Han     Yong Wang

Intel China Research Centre
Intel China Ltd.
Beijing 100020, PRC
Email: jiang.han@intel.com
yong.wang@intel.com

ABSTRACT

In conventional form-based dialogue manager, the task structure of a given task domain is built on E-form or flat slot structure, which some useful structure information of the task domain has been lost. This paper describes a hierarchical task structure model that is domain independent at least for a class of task domains. Based on the task structure model, a rule based dialogue management, which has some domain independent “conversational skill”, is presented.

1. INTRODUCTION

The spoken dialogue system combines speech recognition with natural language understanding, language generation, and dialogue management[1]. It engages the user with a multi-utterance conversation in order to complete a goal-directed task. The successful completion of a task requires that, at the end of a session, the two agents (user and system) involved agree on a particular result. It also requires that, between them, they have some common understanding of how to go about completing the task. This in turn requires that two types of knowledge be captured in systems that successfully supports this task: a representation for the task domain specific information that needs to be created and representation that captures the structure of the activity needed to complete task. It has been suggested that properly devised data structures that reflect the constraints imposed by the task domain can be used to interpret user inputs and guiding dialogue[2][3][4]. However, It is also desirable to capture the structure of the activity that takes place and thereby capture whatever expertise may be involved. This expertise reflects what humans know about performing tasks; it is both domain-specific and more general, representing “conversational skills” that may be transferable between task domains.

Our goal is to provide a spoken dialogue toolkit that make it easier for a developer to build his spoken dialogue application system. To achieve the goal, we will face two issues on dialogue management: providing a general task structure model, and abstracting the domain-independent from the domain-dependent aspects of dialogue manager. Through some empirical studies of dialog process in purposeful task domain such as travelling plan and meeting scheduler, we believe that this kind of dialogue process can be modeled as an approximating procedure with contributions from user and computer as input and converging on task completion. Based on the belief, we have been exploring the two dialogue manager issues discussed above.

Our approach is based on a generalization of the form-filling paradigm. We use a hierarchical task structure model to describe task domain specific information. It is a general-purpose task structure model that can be regarded as domain independent, at least for a class of tasks.

The dialogue management is designed based on the proposed task structure model. It is a semantic data driven dialogue manager in which the current context of the system is used to decide what to do next. Systems do not need a dialogue control table; a general engine operates on the semantic representation and the current context to control the interaction flow.

2. HIERARCHICAL TASK STRUCTURE MODEL

Given a task domain, it contains many pieces of information. The user can speak these information pieces in one utterance in any reasonable combination way. It is our common belief that these information pieces in a given task domain do have their own structure which can help the dialogue manager to interact with user to complete the task. In conventional form-based dialogue manager, the task structure is an E-form or flat slot structure[5], in which some useful structure information has been lost[6]. This results in domain-dependent context merging of dialogue management, because some domain specific context merging rules have been involved into dialogue manager to cope with the interaction of multiple constraints on the same slot or the interaction between the various slots.

Based on the above analysis, we proposed a hierarchical task structure model. Figure 1 illustrates the task structure model. We can see that the model is a hierarchical structure in contrast...
to the conventional flat slot structure. It uses object-oriented method to specify and abstract semantic information in different level. In detail, there are five levels in the model: In frame list level, a frame list object which consists of some frame objects is used to describe a cluster of tasks or topics; In frame level, a frame object which consists of some slot objects is used to describe a task or topic; In slot level, a slot object which consists of some key objects is used to describe concept; in key level, a key object which consists of some tag objects is used to describe sub concept; we call these tag objects as tag cluster which is used to describe information pieces.

![Diagram of task structure model](image)

**Figure 1. Hierarchical task structure model**

The main merit of the task structure model is that we can use some mathematical objects, for instance, partial order structure, set structure or other mathematical structures, to describe the structure of a tag cluster. Since the operational semantics of a mathematical object can be described formally, so the operational semantics of a tag cluster can be instantiated by its corresponding mathematic object. In a great degree, this can benefit the domain-independent design of dialogue management. Also, for the purposeful task domain, the task structure model can be used to represent the converging point of the task completion. On the other hand, based on the model, the slot list object that consists of some slot objects can be introduced to represent user utterance. Obviously, the slot list object has tree structure on the whole. Moreover, in general the parse generated by natural language understanding module such as FSG parser or Phoenix robust parser [1] also has tree structure. This really can help to implement a domain-independent information extractor that will extracts information from the current parse to a slot list object. The slot list will be sent to dialogue management module to do context tracking.

Given a task domain, the general principles to define the task structure of the task domain according to the proposed task structure model are:

- In frame level, define loosely coupled concepts as slots,
- In slot level, define loosely coupled sub concepts as keys,
- In key level, put tightly coupled information pieces or tags into a tag cluster.

The following is a simple example of a task structure specification for meeting scheduler domain:

**Frame_name: Meeting Scheduler**
**Frame_begin**
**Slot_name: time**
**Slot_begin**
**Key_name: time_from**
**Key_begin**
operational semantics: order structure
day_range +e; am_pm +e; hour +e; minute +e
**Key_end**
**Key_name: time_to**
**Key_begin**
operational semantics: order structure
day_range +e; am_pm +e; hour +e; minute +e
**Key_end**
**Key_name: time_duration**
**Key_begin**
operational semantics: order structure
day_range +e; am_pm +e; hour +e; minute +e
**Key_end**
**Slot_end**
**Slot_name: location**
**Slot_begin**
**Key_name: location_place**
**Key_begin**
operational semantics: order structure
room_name +e
**Key_end**
**Slot_end**
**Slot_name: attendee:**
**Slot_begin**
**Key_name: attendee_name**
**Key_begin**
operational semantics: set structure
name +r
**Key_end**
**Slot_end**
**Frame_end**

We can see that there are three concepts (slots) in meeting scheduler domain. They are time slot, location slot, and attendee slot. For time slot, there are three sub concepts (key): time_from, time_to, and time_duration. Here the following two typical tag clusters deserve to be pointed out: < day_range +e; am_pm +e; hour +e; minute +e> in time slot and {name +r} in attendee slot. In the two tag clusters, +e and +r represent the tag’s attribute:

- +e means the tag will occur in a tag cluster just one time,
• +r means the tag can occur in a tag cluster many times.

For the tag cluster < day_range +e; am_pm +e; hour +e; minute +e>, its operational semantics can instantiated by partial order structure, whereas the operational semantics of tag cluster {name +r} can be instantiated by set structure.

In our dialogue tool kit, we developed a task structure specification language to describe the task structure for a given task domain based on the discussed task structure model. For a given task domain, a task structure specification will be written by application developer(s) using the language, which will be used to instantiate the dialogue manager.

### 3. DIALOGUE MANAGEMENT

Our rule based Dialogue Management (DM) is responsible for processing some general “conversational skills”, such as prompting for missing information, clarifying ambiguous information, intention interpretation, handling errors and mutually grounding between human and computer based on confidence measure, according to the task structure model discussed above.

![Figure 2: Dialogue management concept structure](image)

Figure 2 illustrates the concept structure of our DM from abstract level. From this figure, we can see there are four knowledge bases. Each knowledge base has specific function. Also we can see there is one rule base, in which there are some rules to describe the information flow between these knowledge bases in formalized way to process those common knowledge base.

- **Target Base**
  This knowledge base is coming from outside of DM. The base is used to store the task information, including the task structure information that has been discussed above. The base is also used by rule base to determine if the whole dialogue process has reached the converging point of a given task domain.

- **History Base**
  This knowledge base is defined as a container that stores all the semantic data coming from outside of DM in the whole dialogue process, such as from parser, from domain-specific agents, or from DTMF device in telephony application. The base is used by rule base to save all information that comes into DM during the whole process. In other words, we can recover exactly the process of dialogue via browsing the base.

- **Unclear Base**
  This knowledge base is defined as a container that stores the information. The information is extracted from History Base and is required to be grounded between user and dialogue manager. This base is used by rule base to pick up information that waits for confirming via using some rules.

- **Useful Base**
  This knowledge base is defined as a container that stores mutually agreed-upon information between human and computer, and can be regarded as an approximation point that reflects in-time status of dialogue process. The base is used by rule base to pick up information from History Base and can be used to recover in-context meaning from the surface semantics of outside input. It can be seen as the currently active context information.

The Rule Base abstracts the domain-independent from the domain-dependent aspects of DM, and describes some general “conversational skills” of DM. According to the function of each rule in Rule Base, it can be classified to three rule classes:

- **Rule class for outside-input information filtering**
  This rule class determines what is good information for the given task domain. We can filter some task-unrelated or over-informative information from outside input using this rule class. All the information extracted by filter satisfies the following conditions: It should be the newly coming outside-input information; it should be helpful to solve task. In summary, the rule class controls the information flow from History Base to Unclear Base, based on Target Base.

- **Rule class for information usefulness decision**
  This rule class determines what information can really be used to solve task, what information has been grounded by user and computer. Generally speaking, the rule class controls the information exchange between Unclear Base and Useful Base.

- **Rule class for task completion decision**
  This rule class determines whether the task has been completed. In detail, the rule class checks the Target Base and Useful Base to determine if the two bases are matched with each other. If matched, it will capture the converging point of a given task and trigger a task end processing procedure. If not, it will figure out what DM does not know yet, and trigger a
Based on the dialogue management concept structure, we make use of the following algorithm to do context tracking:

1. **Extract information**
   The main function of this part is extracting information from the current parses, the information is generated by natural language understanding module with semantic tags of tree structure.

2. **Combining extracted semantic information**
   The main function of this part is to do merge with the context meaning.

3. **Interacting with backend resources**
   The main function of this part is to do interaction with backend server based on the combined context when DM finds it is necessary to do so.

4. **Updating the context**
   The main function of this part is to update the context based on the interacting with backend resources.

5. **Producing a response**
   The main function of this part is to generate the summary of the current dialogue process status and prompting for more information to user if necessary.

The dialogue manager based on proposed task structure model is domain-independent, at least for a class of task domains. The main reason is that the operational semantics of the lowest level objects (tag clusters) in the proposed task structure model can be modeled as that of mathematical objects, so it can be formalized. Therefore, the context tracking of the proposed dialogue management architecture can be implemented purely domain independent. Also the architecture is open; we can add new discourse-level knowledge to the architecture though adding some relevant rules.

### 4. SUMMARY AND FUTURE PLANS

Our goal is to provide a spoken dialogue toolkit that make it easier for a developer to build his spoken dialogue application system. Domain portability is our key consideration. We proposed a hierarchical task structure model to capture intrinsic information of a given task; Based on the task structure model, a rule based dialogue management is presented. Based on the two techniques, we implement an initial version of spoken dialogue tookit. Using the toolkit, application developer just need to prescribe the task domain itself in a simple declarative manner through some specification files, then the instantiated dialog manager will be cooperative with the user to solve the task in mixed initiative way. To test and improve the tool kit, we built some experimental spoken dialogue systems such as meeting schedule, auto attendant, email checking, TV program guiding and so on. The experimental results show that the domain portability of the tool kit satisfies the design goal, but it lacks of enough flexibility to let developer to customize his spoken dialogue system in his own way. Also, the task structure model should be improved. We plan to provide more flexible way for user to specify the task structure, not only in tree structure, also in graph structure to allow high-level concept being shared. This is necessary for some complicated application such a call center.

### 5. REFERENCES


