

國立交通大學

電控工程研究所

碩士論文

對於祈使機械控制命令的理性預期模型設計

Designing a Rational Expectation Model for Imperative Machine

Control Commands



研究生：黃晉澤

Student: Jin-Ze Huang

指導教授：黃育綸 博士

Advisor: Dr. Yu-Lun Huang

中華民國一百年七月

July, 2011

對於祈使機械控制命令的理性預期模型設計

Designing a Rational Expectation Model for Imperative Machine Control

Commands

研 究 生：黃晉澤

Student: Jin-Ze Huang

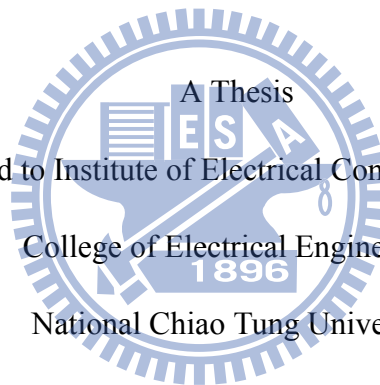
指導教授：黃育倫 博士

Advisor: Dr. Yu-Lun Huang

國 立 交 通 大 學

電控工程研究所

碩士論文



Submitted to Institute of Electrical Control Engineering

College of Electrical Engineering

National Chiao Tung University

in partial Fulfill of the Requirements

for the Degree of

Master

in

Institute of Electrical Control Engineering

July, 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年七月

Designing a Rational Expectation Model for Imperative Machine Control Commands

Student: Jin-Ze Huang

Advisor: Dr. Yu-Lun Huang

Institute of Electrical Control Engineering

National Chiao Tung University

Abstract

自然語言處理的相關研究可以追溯到1950年代，為了讓一般使用者能夠用自然語言與電腦系統溝通，許多研究學者投注於基於自然語言之對話系統的研究。現有的研究依其設計，主要可以分為三種基本的對話方法 - 基於有限狀態（Finite State Machine）、基於框架（Frame），以及基於資訊狀態更新（Information State Update）等對話方法。在這篇論文中，我們定義了祈使控制機械命令，用來組成一個任務，一個任務包含了受控元件、行為、受控元件的描述。我們提出一個理性期望模型，結合有限狀態機與分散式代理人等設計概念，分析我們所定義的任務。其中，有限狀態機是用來處理任務中的受控元件、行為、描述子等關係，簡化後續對話方法中各分散式代理人（包括元件代理人、行為代理人、回應代理人、命令代理人等）所負責的處理程序。最後，我們並以各種不同任務分析，結果可以明顯看出本文所提之方法可以幫助對話管理者減少對話的次數。

Designing a Rational Expectation Model for Imperative Machine Control Commands

Student: Jin-Ze Huang

Advisor: Dr. Yu-Lun Huang

Institute of Electrical Control Engineering

National Chiao Tung University

Abstract

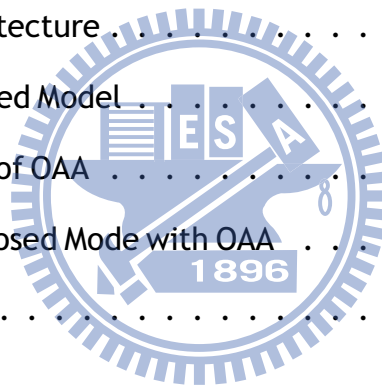
Since 1950s, natural language processing and dialog approaches, concerning the interactions between human users and computers, have been researched and evolved towards a more friendly user interface. By their designs, the existing dialog approaches can be classified as Finite State-based, Frame-based and Information State Update (ISU)-based. In this thesis, we define imperative machine control commands to construct a task with operated device , behavior and description of the device. we propose a rational expectation model, integrated with a finite state machine and distributed agents, to analyze the task we defined. A finite state machine is designed to construct relationships of a operated device , behavior and description describing the device. Such a design simplifies the process of distributed agents, including the component agent, behavior agent, response agent and command agent, in a dialogue system. In the end of the thesis, we analyze REM in terms of different case studies of tasks. From the case studies, we show that the REM can decrease the conversation of dialogue manager.

Contents

Abstract	i
Abstract	ii
Table of Contents	iii
List of Figures	vi
Chapter 1 Introduction	1
1.1 Dialogue System	1
1.2 Approaches of Dialogue Manager	3
1.3 Motivation	4
1.4 Contribution	5
1.5 Organization	5
Chapter 2 Background	6
2.1 Approach of Dialogue	6
2.1.1 Finite State-Based Approach	6
2.1.2 Frame-Based Approach	9
2.1.3 Information State Update Approach	12
2.2 Current Dialogue System	14
2.2.1 FSM-based Dialogue Systems	14
2.2.2 Frame-based Dialogue Systems	15
2.2.3 Information State Update Dialogue System	17
2.3 Summary	17
Chapter 3 Design	19



3.1 Preliminaries	19
3.2 Definition of Task	21
3.3 Block Diagram	21
3.4 Flow Chart	25
3.5 Infer missing information	30
3.6 Summary	30
 Chapter 4 Implementation	 35
4.1 Environment	35
4.1.1 Hardware	35
4.1.2 Software	35
4.2 Open Agent Architecture	36
4.2.1 Agent-Based Model	36
4.2.2 Structure of OAA	37
4.3 Structure of Proposed Mode with OAA	38
4.4 summary	39
 Chapter 5 Analysis	 41
5.1 Case Study	41
5.1.1 Case 1	42
5.1.2 Case 2	43
5.1.3 Case 3	44
5.1.4 Case 4	45
5.2 Discussion	45
5.2.1 Finite-state approach with REM	45
5.2.2 Frame-based approach with REM	46
 Chapter 6 Conclusion	 47



Chapter 7 Future Work

48

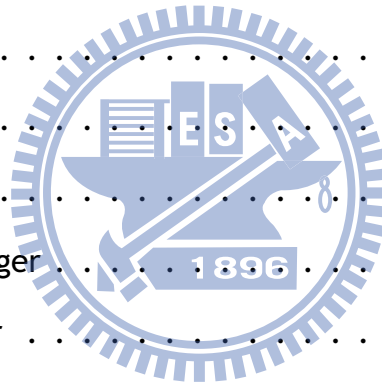
References

49



List of Figures

1.1	Dialogue System	2
1.2	Architecture of Dialogue System	2
2.1	Finite State-Based approach for component in home control	7
2.2	Frame-Based Model	10
2.3	Exmample of Finite-based Dialogue System	16
3.1	Parse Tree	20
3.2	Block Diagram	22
3.3	Major Flow	25
3.4	Overview	27
3.5	Task Generator	31
3.6	Component Manager	32
3.7	Behavior Manager	33
3.8	Response Manager	34
4.1	Agent-Based Model	37
4.2	Open Agent Architecture	38
4.3	Structure of Proposed Model with OAA	39
4.4	Facilitator with One Manager	40
4.5	Facilitator with Two Managers	40



Chapter 1

Introduction

This chapter introduces the definition and approaches of a dialogue system, and then describes our motivation and contribution.

1.1 Dialogue System

The major goal of a dialogue system is that a user can use interact with the dialogue system by natural language, as shown in Figure. 1.1. The research can be traced back to Artificial Intelligence(AI) in 1950s, a number of research has addressed this goal until now. In general, the dialogue system can help user complete some tasks in specified domain.

For example, a user may control home appliance by simple words or sentences. Or, a driver who is in car wants to play music, go home by navigation system and call someone within driving. A user can use natural language as command, the dialogue system should realize the semantic from user. So, the dialogue system should analysis the sentence from human to realize semantic.

There are several components in the dialogue system, as shown in Figure. 1.2 We take a flow to explain the integral system. The input is a sentence with speech and the dialogue manager commands the speech recognition component recognizes the sound sentence to text. The semantics parsing component receives the text from speech recognition and analyses the sentence to realize the semantic from a user. The semantics parsing component parses the result to dialogue manager. The result may be a controlled command or unmeaning sentence.

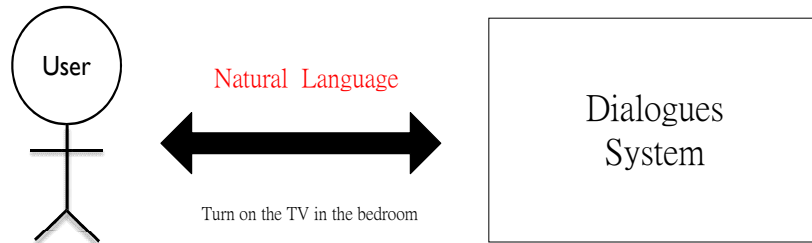


Figure 1.1: Dialogue System

The dialogue manager sends the command to the external communication component according to the result as controlled command. The external communication component is responsible for communicating with external device and controlling the devices. At last, the speech synthesis component transforms the texts to voice and outputs the voice to user.

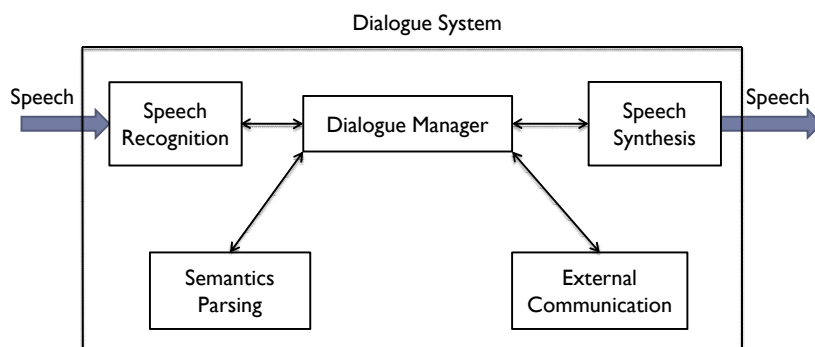
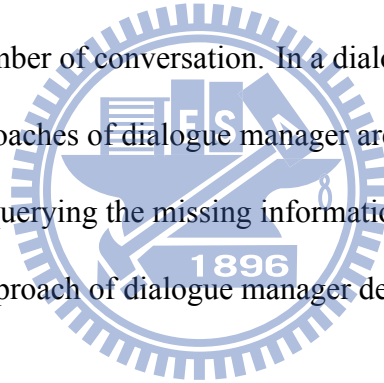


Figure 1.2: Architecture of Dialogue System

We take an example in home controls, if a user wants to control the television in the

bedroom. User may say "Turn on the TV in the bedroom.". Then the speech recognition component recognizes the sound sentence to text. The semantics parsing component realizes the semantic, let dialogue manager know user's command. The dialogue manager will receive the information, and connect other components according to the semantic of user's command. The dialogue manager requires the external communication component to communicate the device "TV" and send the command "turn on" to device. The response generation component creates the response according to result of external device as "complete" or "another command". The speech synthesis component transforms the response to speech and outputs to user. User will hear the sound response not just text.

For missing information, the dialogues manager would ask the user for missing parts, and the process increases the number of conversation. In a dialogue system, the dialogue manager is the system core. The approaches of dialogue manager are responsible for the flow of conversation, including for querying the missing information. The dialogue manager controls the dialogue flow and the approach of dialogue manager decides the dialogue type.



1.2 Approaches of Dialogue Manager

Dialogue system can be classified to three approaches. We will introduce these approaches in Section 2.1 : a. **Finite state-based approach** - A sequence of predefined questions for a task. b. **Frame-based approach** - A set of predefined condition for a task. c. **Information State Update approach** - A number of sets of conditions and effects support the information state changing.

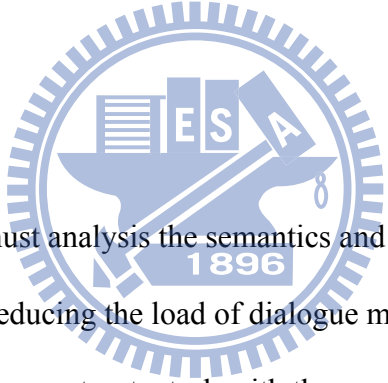
We describe the three approaches shortly.

- In finite state-based approach, the dialogue system controls the flow of conversation

with a sequence of predetermined questions. A general system with state-based dialogue constraints the user's input. Because the restriction of input, the system can be constructed easily.

- In a frame-based approach the user is asked questions that enable the system to fill slots to perform a task. The dialogue flow is not predetermined but reliable for user's input.
- Information State Update approach (ISU) [1] is similar with frame-based approach, but more complex. The major difference between ISU and frame-based is that ISU has a information state to update with rules. A programmer must determine the a set of dialogue move, a set of update rules and an update strategy.

1.3 Motivation



The dialogue manager must analysis the semantics and resolve the missing information by mentioned approaches. For reducing the load of dialogue manager, we determine imperative machine control commands to construct a task with the operated device, its descriptions and behaviors. Moreover, we design its rational expectation model (REM) for missing information between the semantics parsing component and the dialogue manager.

The imperative machine controlling commands would compose a task and assist REM to check the integrity. We apply finite-state machine and the concept of distributed agents for designing REM. REM would inspect the integrity of the task step by step, and deal with the missing information of the task by distributed agents.

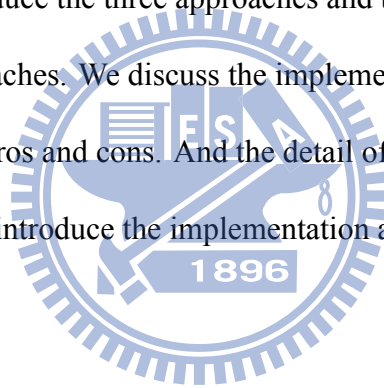
We design a set of distributed agents to deal with the missing information in a task dependently. We use Open Agent Architecture (OAA) [2] to implement the distributed agents, we would introduce the OAA in Section 4

1.4 Contribution

We expect that imperative machine controlling commands and REM would reduce the load of the dialogue manager. For determining the form of the controlling command, the dialogue manager realizes the semantics easily. REM deal with missing information before the dialogue manager. The dialogue manager does not handle the missing information by asking users or querying the database.

1.5 Organization

In Section 2.1, we introduce the three approaches and their principle advantage and disadvantage of those approaches. We discuss the implementation of the approaches in Section 2.2 and show their pros and cons. And the detail of REM is revealed in Section 3. In Section 4 and Section 5 we introduce the implementation and discuss all scenarios. Section 6 concludes the thesis.



Chapter 2

Background

In the chapter, we talk about background of our research and current dialogue system.

2.1 Approach of Dialogue

There are three basic strategies for a dialogue system -- finite state-based approach, frame-based approach [3] and information state update (ISU) approach respectively. We discuss pros and cons of finite state-based approach, frame-based approach and ISU approach.

2.1.1 Finite State-Based Approach

In a basic finite state-based system, the dialogue structure is represented in the set of a state and possible transitions. The states in the system represent information from user, and the transitions between nodes determine possible path. The system progresses through a series of states, with the transitions between nodes which are determined by the user's input. In finite state-based approach, the system consisted of a sequence of predetermined question and corresponding keywords, as show in Figure. 2.1. All paths and questions are predefined.

Most commercial systems are implemented with this approach, because the dialogue flow is specified as a sequence of states with possible transitions. The system maintains the transitions of the dialogue by recognize the answer of question, as words or phrases of user's responses.

We take an example about home control. The example of specified flow of a dialogue system which verifies the user's response of each state. There is a assumption that accuracy rate of the speech recognizer is 100%.

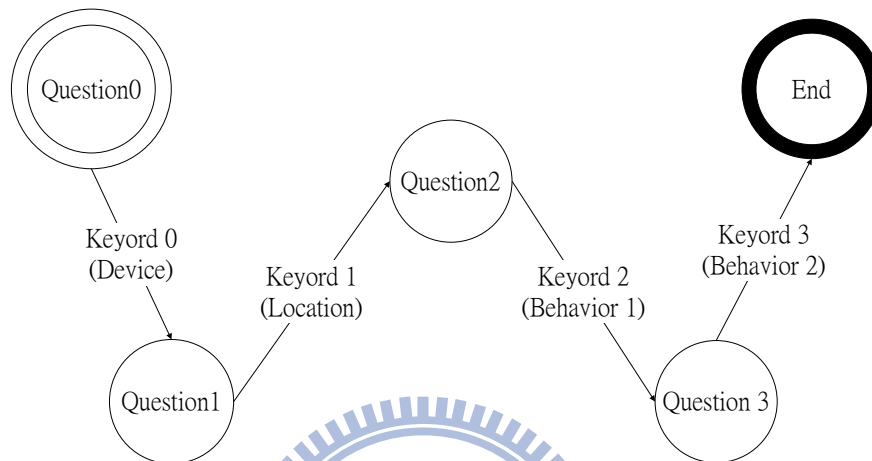


Figure 2.1: Finite State-Based approach for component in home control

System:Which one do you want to operate?	[Question 0]
User: The television in the bedroom.	[Keyword 0]
System:Which one television?	[Question 1]
User: In the bedroom	[Keyword 1]
System:What is the instruction	[Question 2]
User: Turn on and go to channel 36	[Keyword 2]
System:What is the next instruction	[Question 4]
User: Go to channel 36	[Keyword 4]
System:What is the next instruction	
User: No	

In this conversation, system would ask a sequence of questions for the controlled device and behaviors. User's answers have to satisfy the transition condition or system would ask the same question until the corrected keywords occur. And system can not handle over-information, if user provided too much information. The system with finite-state approach can not accept the over-information.[4] And a problem about the orderless of behaviors occur. If a user gives "Go to the channel 36" command before "Turn on the TV" command, the system with finite-state approach can not execute the command exactly.

Generally the system with finite state-based approach restricts user's input to a word or phrase. Each state which receives a predetermined word or simple phrase makes language understand more easier. However, this advantage will make system easy to construct step by step. The flow of dialogue is passive for user, because the restricted input. Even the input is natural input, like sentences, the system must extract the words or phrase in the sentences by the speech recognizer.

In 1996, B. Hansen described a toolkit with typical finite state machine that automatically generates prompts in a variety of styles.[5] The toolkit provides several styles for format of question, so a programmer can design a set of questions with selected styles. A programmer can construct a system by a series of predefine questions.

Advantages

Principal advantage of the finite state-based approach are its simplicity and its execution efficiency. For a developer, the predetermined transition network can approach well-constructed task involving question-response conversation. The well-construed task involves many aspects, as ordered states, information of each state will be dependent or clear transition network. The constructed system guides the flow of dialogue and decides the next

question.

Moreover, the user's responses are restricted, so technological demands will be reduced, particularly the speech recognizer. But the lack of flexibility is a trade-off against the natural input. If the dialogue system is built-in in mobile device, the technological demand is a big issues for implementation. For those reasons, the most commercial systems are implemented by finite-state based approach. As mentioned example, the developers define a series of states and transition network to complete a well-constructed task.

Disadvantages

Finite state-based approach is not suitable for less well-structured task involving unpredicted conversation between user and system. The less well-constructed task means all information of each state will be dependent on the transition network will change within runtime. The dependence means that current state needs information of next state.

The simple example of changing network is that the entry point of system changes or is unpredicted every time. Because the system with finite-state based approach has fixed transition network, the entry point must be unchangeable. It's not natural for user, because user gives the predetermined keyword what dialogue system needs in order. So, The system can't handle additional information in one conversation at a time. For user, the dialogue system with finite state-based approach is not flexible for general lifestyle.

2.1.2 Frame-Based Approach

A basic frame-based system would ask user questions that enable the system to fill slots in a template in order to perform a task. In frame-based approach, the dialogue flow is not determined but dependent on user's input. That means the frame-based approach is

user-initiative because user provides information on user's own initiative.

The questions or tasks of frame-based approach are dependent on their preconditions. The precondition is the information from user, that is the answer of the question. In the Philips timetable system[6], there are predefined conditions that compose the task or question.

In fact, the frame-based approach can import mathematical model for decision making. In 2005, Chin-Han Tsai proposed a dialogue strategy [7] with SA-Q learning [8] and Markov decision [9] for navigation.

We take two examples about home control. The two example of specified conditions of a dialogue system with frame-based approach, as show in Figure. 2.2. There is a assumption that accuracy rate of the speech recognizer is 100%.

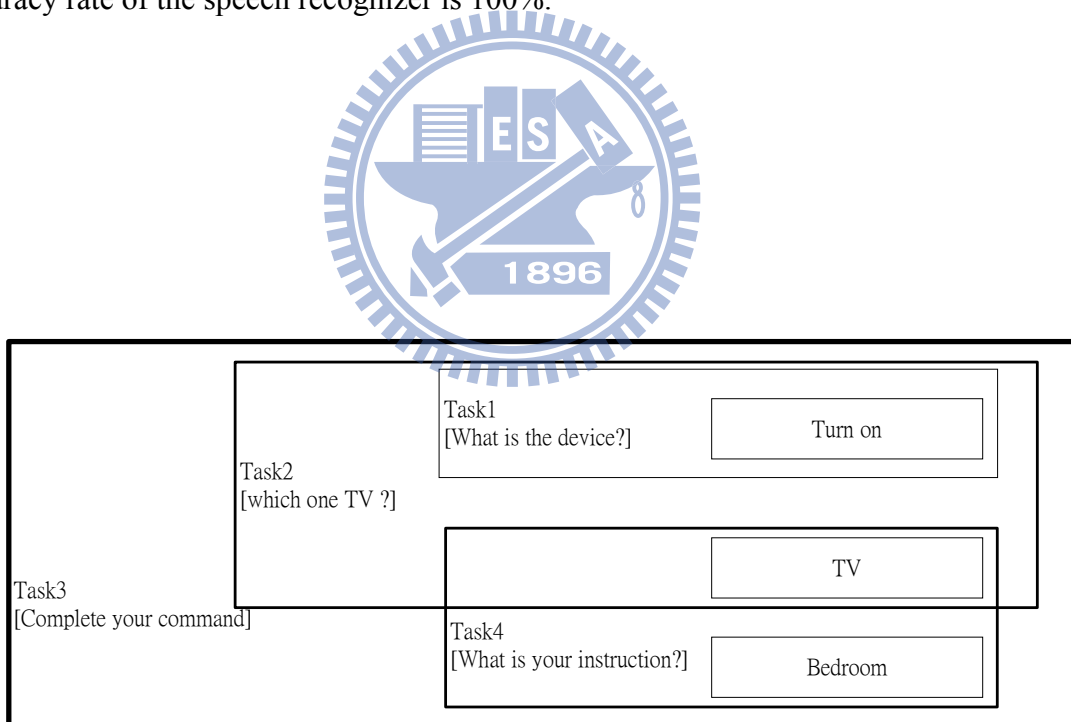


Figure 2.2: Frame-Based Model

Conversation 1

System:What is your command?

User: Turn on the TV. [Task 2]

System:which one TV? [Question 2]

User: In the bedroom

System:Complete your command [Task 3]

Conversation 2

System:What is your command?

User: The TV in the bedroom. [Task 4]

System:What is your instruction? [Question 4]

User: Turn on

System:Complete your command [Task 3]

In the two conversations, the system with frame-based approach would ask a common question at first. A user could say any imperative sentence to be a command, and then the system would find the correct keywords in user's command. The system fills the slots (or satisfies the conditions) to complete some tasks.

Obviously, the system can handle over-information under some circumstances, if a user provided more information. But, if a programmer did not take into account some situation, the system does not complete the task. In summary, the precondition determine the flexibility of the dialogue system and how system executes actions.

Advantages

The frame-based approach has several advantages over the finite-state-based approach. For users, frame-based approach is greater flexibility and the ability of using natural language as input. It is difficult to constrain user's responses required by the system, even when the system have been carefully designed.[10] The user can provide over-information under the frame-based approach. In this way, the transition time can be reduced, that results in a more efficient and more natural dialogue flow.

Disadvantages

Finite state-based approach and frame-based approach are appropriate for well-defined tasks. In frame-based approach, all task should be decomposed to several meaningful slots (or conditions). And only well-defined tasks can be decompose.

In this context, the determination of the system's next action is fairly limited, The developers defined the time of occurrence of tasks (or questions) according to the sets of chosen conditions. So, the frame-based approach is short of scalability.

2.1.3 Information State Update Approach

The ISU approach consists of five concepts.

- **Informational components**, including aspects of common context and internal motivating factors.
- **Formal representations** of informational components.
- A set of **dialogue moves**, that trigger the update of information state.
- A set of **update rules**, that govern the updating of information state.

- An **update strategy**, that deciding which rule(s) to apply.

In a sense, the ISU approach is similar with the frame-based approach. There also are conditions and rules in the ISU approach. But, there is a difference between the ISU approach and the frame-based frame. There is **information state** in the ISU approach, the dialogue state can be represented by information state.

Moreover, a programmer must define the informational components in formal representation. The informational components can be private information or public information. Further, the dialogue moves serve as triggers to update the information state. The set of dialogue moves would influence the possible messages that can be sent and the update to be made.

At last, the programmer should design a set of update rules and choose a update strategy. The update rules formalize the way that information state is changed. There some types of the update strategy, take the first rule, or apply each rule in sequence etc.

Advantages

The ISU approach has several advantages over the frame-based approach. The ISU approach provides the information state to present the dialogue state. The rules can be more complex than the frame-based approach.

The ISU approach is more flexible than frame-based approach, because the dialogue moves can make system be mixed-initiative. The dialogue moves can determine the next information state according to the previous information state and user's input.

It is also difficult to constrain user's responses required by the system, even when the system have been carefully designed. The ISU approach can update a set of informational components in a conversation.

Disadvantages

The disadvantages of ISU approach are how to define the informational components to represent the information state, and how to design the conditions and effects for rules. In a word, the ISU approach is more complex than frame-based model Because the levels of ISU approach is more complicated.

2.2 Current Dialogue System

In the section we would take a look at the few implementations of dialogue system by a or finite-state based and frame-based strategy.

A number of researchers have focused on single strategy for implementation. Some simple application is still constructed by finite-state based strategy, as telephone booking system. The demand of telephone booking system is easily structure, so the system was constructed by finite-state based strategy.

On the other hand, the frame-based strategy will be applied to database query system mostly, as voice navigation system. The voice navigation system will analysis user's instruction, as "Take me to NBA store on 5th avenue.". The system fill in slots of the specified task. The user's instruction will be taken apart to several pattens, as verb or destination. The system would complete the user's request based on the result of decomposing the user's input.

2.2.1 FSM-based Dialogue Systems

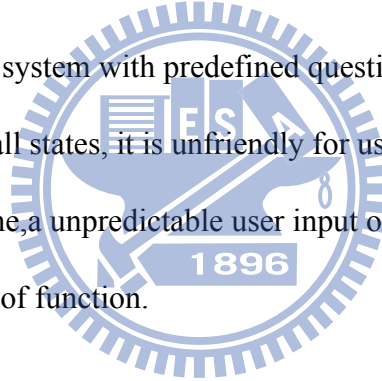
Since 1992, the Center for Spoken Language Understanding (CSLU) developed the CSLU Toolkit [11] as a complete system , including speech recognition and authoring tools etc. By CSLU Toolkit, we can use Rapid Application Developer (RAD) to build real-world dialogue

system. A programmer can define activities by placing objects and assign transition of activities. In 1999, Michael F. McTear provided practical experience for undergraduate students in the specification and development of spoken dialogue systems.[12]

We take a simple ticketing system as explanation according theory of finite state machine. [13] We define a set of questions as a finite set of state $S =$ *Departure, Time, Destination, End* . And we define the transition, as shown in Figure 2.3.

At first, the reservation system asks the user for destination. If the ticketing system does not get the exact answer from user, the system holds the state *Departure* till the correct answer. If the ticketing system receives the correct answers of three states *Departure* , *Time* and *Destination* , the system would enter the state *End*.

Obviously, the ticketing system with predefined questions is lack of flexibility. And user must complete condition of all states, it is unfriendly for user. Even programmers design a integrated finite-state machine, a unpredictable user input or unpredictable behaviour of devise will make the system in loss of function.



2.2.2 Frame-based Dialogue Systems

In 1995, "The Philips Automatic Train Timetable Information System" [6] provides information about train connection between German cities. And in 2000, "MIMIC: an adaptive mixed initiative spoken dialogue system for information queries" [14] provides movie showtime information. The two system had the same aim, that is how to construct an appropriate database query that user required. In fact, the Philips system still is user-initiative, the MIMIC is mixed-initiative by modelling initiative during dialogue interaction [15]

The task specification in MIMIC consists of four slot, Question-Type, Movie, Theater and Town respectively. The MIMIC use goal selection algorithm to determine the action of system

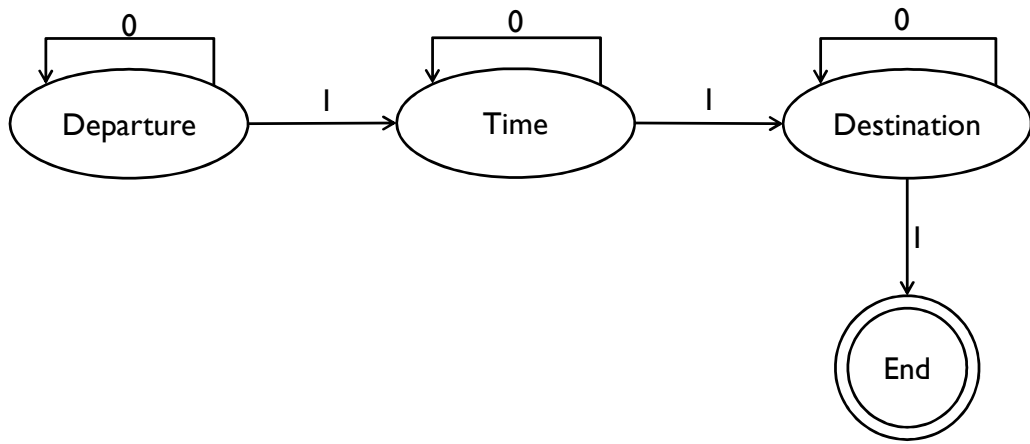


Figure 2.3: Example of Finite-based Dialogue System

to find the goal with basic probability (bap). The MIMIC would update the baps in runtime with a set of initial value.

Obviously, the frame-based strategy would be suitable for well-defined task. If a task has unknown number of slot, e.g controlling a device, the frame-based strategy will be complex. We would say a series of instructions to a device. For handle the situation of unknown number of slots, developer must design more combinations of slots for tasks. So, the frame-based strategy is similar with finite state-based strategy in a aspect of weak ability of unknown situation.

2.2.3 Information State Update Dialogue System

In 2007, Amores et al. [16] [17] proposed a multimodal and multilingual dialogue system for the home domain (MIMUS). MIMUS follows the Information State Update approach to dialogue management, and has been developed under the EU-funded TALK project [18].

MIMUS

Architecture of MIMUS is a set of OAA (Open Agent Architecture) agents [2]. The system core is **Dialogue Manager**, which processes all requests from other agents, the user's input and provides appropriate output. Information transformation between all agents is controlled by system core.

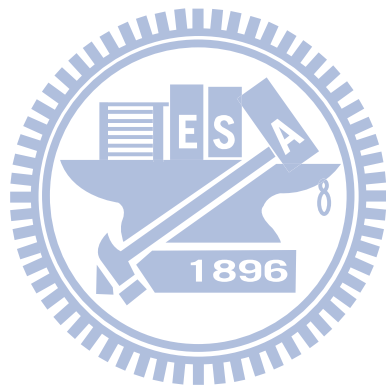
Because of Open Agent Architecture, every manager will complete subsection of user's request. And communication of every manager still pass through the dialogue manager, every manager will send the result of subtask to the dialogue manager. The main approach implementing the system is Information State Update (ISU). The principal element of ISU approach is the dialogue history, which memorizes dialogue states and is updated by some update rules.

The informational components in MIMUs are **Dialogue Move, Type, Arguments** and **Contents** (DTAC) [19]. The DTAC obtained for a keyword or a phrase trigger the dialogue update rule.

2.3 Summary

In summary, all approaches have their own advantages and disadvantages, and there are implementation with those approaches. Some are more easy to construct, some are more friendly for users. But, the implemented systems must query the missing information from

asking users.



Chapter 3

Design

3.1 Preliminaries

In the section we will define the notation of controlling commands. First We were inspired by Phrase Structure Grammar (PSG), that is a grammatical notion presented by Chomsky in Syntactic Structures (1957) to represent the structure in language phrases.[20] Based on PSG, we could decompose the sentence to noun-phrase as NP and verb-phrase as VP. In general, the VP will contain NP and verbs and the NP will contain nouns.

Further, we define some notation . (Table 3.1) We define the device that receive user's command is dominated device (C_x). And general noun (N) often means the operated-state of dominated device. User's commands map to the behavior (b_{xi}) of the dominated device. The subscript x is corresponding to the operated device (C_x). The subscript i is a index means order of behaviors. The subscript x is corresponding to the operated component (C_x). We define the description as D_x that includes adj and Loc .

Table 3.1: Notation of Cluster

Object	Notation
Dominated Device	C_x
General Noun	N
Behavior (Command)	b_{xi}
Location	Loc
adjective	adj
Description	D_x

Table 3.2: Notation of Set

Sets	Notation
Set of C_x	\tilde{C}
Set of b_{xi}	β
Set of C_x, D_x, b_{xi}	$Task$

Table 3.3: Notation of Example

TV	C_x
Turn on	b_{tv1}
go to	b_{tv2}
red,room	D_x
b_{tv1}, b_{tv2}	S_{tva}

And we define the set of notation. (Table 3.2) As mentioned PSG, the structure of the sentence is often drawn out as a parse tree, shown in Figure 3.1. Due to the parse tree, we can obtain the structure of sentence easily. Give a example "Turn on the red TV in the room and

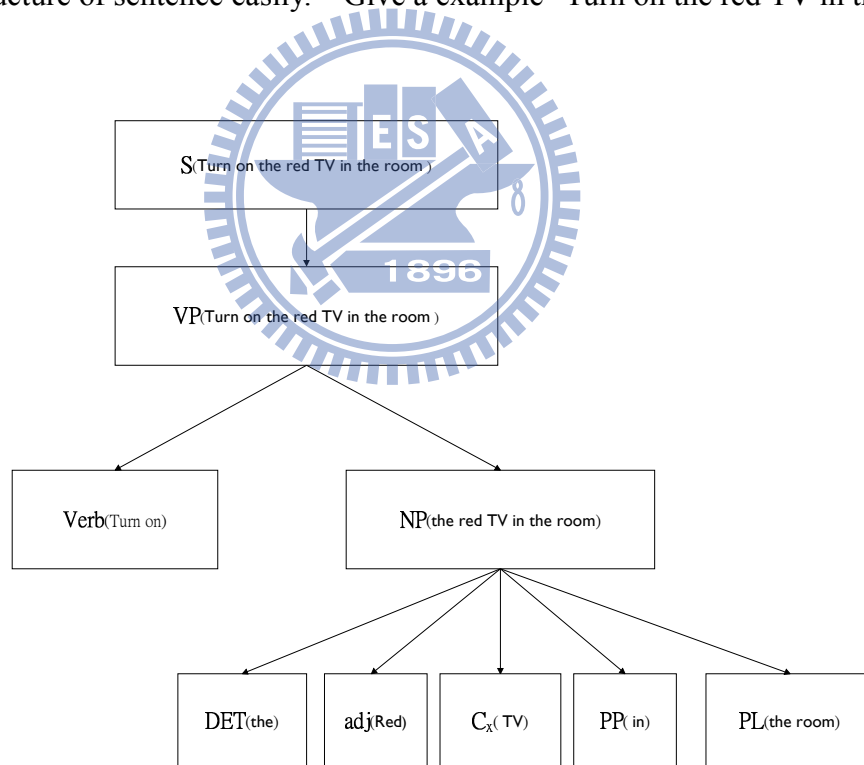


Figure 3.1: Parse Tree

go to channel 36". As mentioned, we can disassemble the sentence to several parts by PSG. First, the sentence will be taken apart to two VP_s according to the verbs "Turn on" and "go to". Second, the VP_s are taken apart to several segments. (Table 3.3)

3.2 Definition of Task

We define a task including three slots, as C_x , b_{xi} and D_x . A task would represent an imperative machine control command, including the elements we define before. The user's sentence would be transformed through semantics parsing, and the task generator, we introduce in next section, would generate the task with the result.

A task should include C_x or b_{xi} at least, for example, a C_x could compose a task. The composition of a task is limited for three elements, but the number of b_{xi} and D_x is not limited. A task would include more than one behavior (b_x), for an example, task = {*tv,turn on,go to the channel 36,bedroom*}. The mentioned task means the operated device *tv* would receive more than one command, as *turn on* and *go to channel 36*.

But, the situation about missing information would happen unexpectedly, for an example, task = {*tv,bedroom*}. The mentioned task is lack of behaviors, the sub-model in Section 3.3 could analysis the task, and deal with the missing information in the task.

3.3 Block Diagram

Abbreviation In the section we discuss out block diagram in REM, and the architecture of the model we proposed is according to OAA [2]. All sub-models in the model are agents that be responsible for different function based on the theory of distributed agents. So, every agent is responsible for of the task. Here is the block diagram, as shown in Figure. 3.2.

We would introduce function of the sub-models in REM, and flow chart of the sub-models.

Component Sub-model The facilitator receives the tagged word and dispatching a part of the task to the suitable agent according to the label. The tagged words as information of the

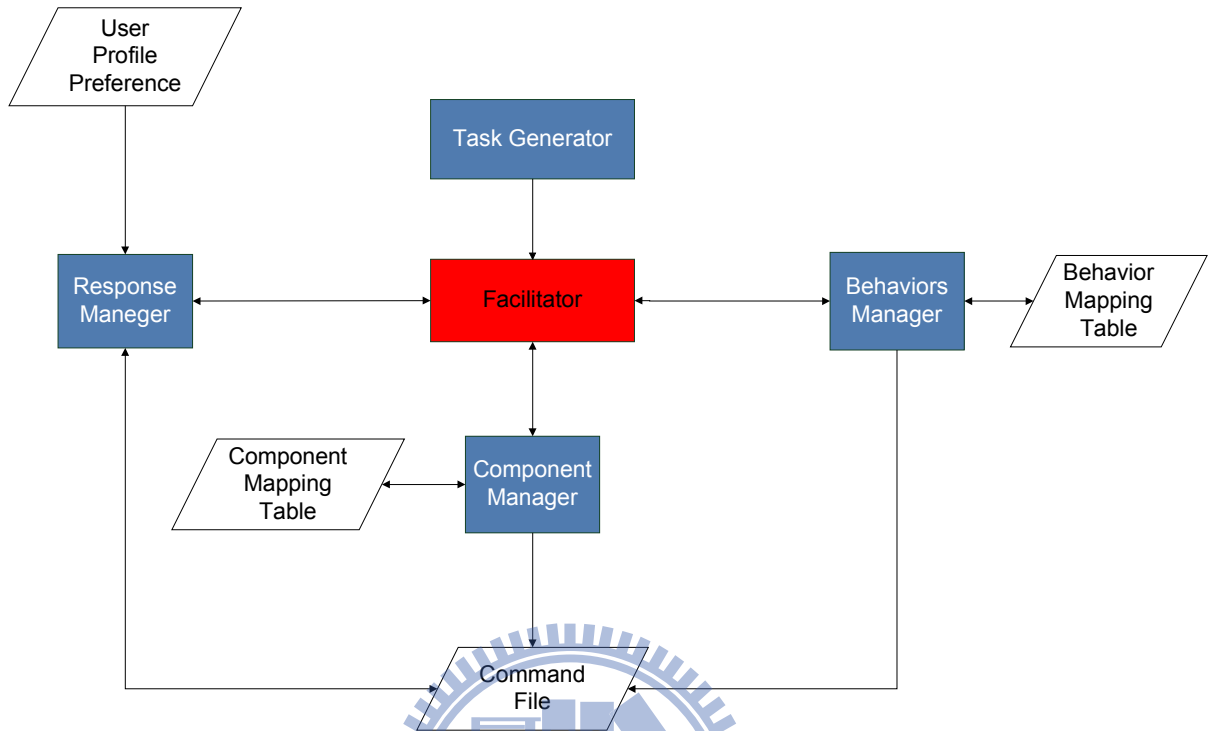


Figure 3.2: Block Diagram

component including descriptions are stored in memory, and the facilitator sends the words to the component sub-model. The component sub-model fills actual command in instruction file. There is a component mapping table to support component sub-model mapped the user's command to actual instruction. The component sub-model can handle the unknown device according to some rules. For example, if we want to add a new TV in system, the NLTK can identify the word "TV" and the word "TV" has a device type code, then the component sub-model can update the component mapping table. The component sub-model will analysis the description of the device ,including the location and some additional feature of device. The description help the component sub-model find the exact device. For example, there are more than one TV in the house but in different room, as bedroom or living room. The component sub-model would find the exact device according to description in user's command. And all

descriptions of device still have mapping code of actual instruction in the component mapping table.

Behavior Sub-model The tagged words as instructions of the device will be stored in memory, and the facilitator sends the words to behavior sub-model. The behavior sub-model fills actual command in instruction file. There still is a behavior mapping table to support the behavior sub-model mapped the user's commands to actual instructions. The behavior sub-model can identify the whether user's command is suitable for device or not. For example, we can't let TV dry clothes, because the command "Dry" should belong to washer. And the behavior sub-model fills a set of instructions in command file in order according to the sequence of user's command. The behavior sub-model also can build some relationship between new behavior and device.

Missing Information Sub-model The missing information sub-model handles the lack of the user's command. There are several possible scenarios. First, all information of user command is integral, the missing information sub-model will not do anything. Second, there is lack of information, the missing information sub-model adds some instruction based on user preference. Because the user preference includes location and costumed set of user.

The missing information sub-model can find the exact device with the location of user if user did not give the completed information of device. For example, user is in the bedroom and user want to turn on the TV in the room. But the user may just say "Turn on the TV". The missing information sub-model will find the TV in the bedroom according to the location of user.

And we design the mapping table with relationship between b_{xi} and C_{xi} .

Component Mapping Table The component mapping table helps component sub-model fill the actual instruction in command file. We design the component mapping table with three columns. device ID location ID feature ID

The first is device type with two fences, the second is location with two fences and the last is feature with four fences. The feature column can be costumed by user or programmer.

For example, the feature column can be the brand of device,color of device or more details of device, even be combination of brand and color. The component sub-model should maintain the mapping table. As mentioned above, the component sub-model updates the mapping table by some logical rules.

Behaviour Mapping Table The behavior mapping table will help behavior sub-model fill the actual instruction in instruction file. The behavior mapping table has three columns. device ID state ID parameter weight

The first is device type with two fences, the second is state of device with two fences and the last is parameter with four fences. The parameter column is the follow-up parameters of the user command. The last column is weight of the device, based on the users trends.

For example, the user may watch the channel 73. The TV will receive the instruction of changing channel and the number 73 is the follow-up parameter of the command. The weight of TV would increase when the task include the C_x TV.

The behavior sub-model should maintain the mapping table. As mentioned above, the behavior sub-model updates the mapping table by some logical rules.

In fact, the three sub-models can handle most scenery in missing information of tasks. And we add some user information in system to enhance the usability of the proposed sub-models.

3.4 Flow Chart

Flow Chart We apply finite state machine to major flow with the task generator determines determining weather the C_x , D_x and b_{xi} in the sentence or not, and with the missing information sub-model, behavior sub-model and component sub-model, as show in Figure. 3.3 The notation [!] means the element is non-existent in sentence.

At first, we check C_x and D_x whether in sentence, and then check the b_{xi} . If there is not any C_x , D_x or b_{xi} in sentence, the **Behavior sub-model** and the **comPonent sub-model** handle the lack of information. If user provides complete information in sentence, the **Missing Information Sub-model** handle the new elements.

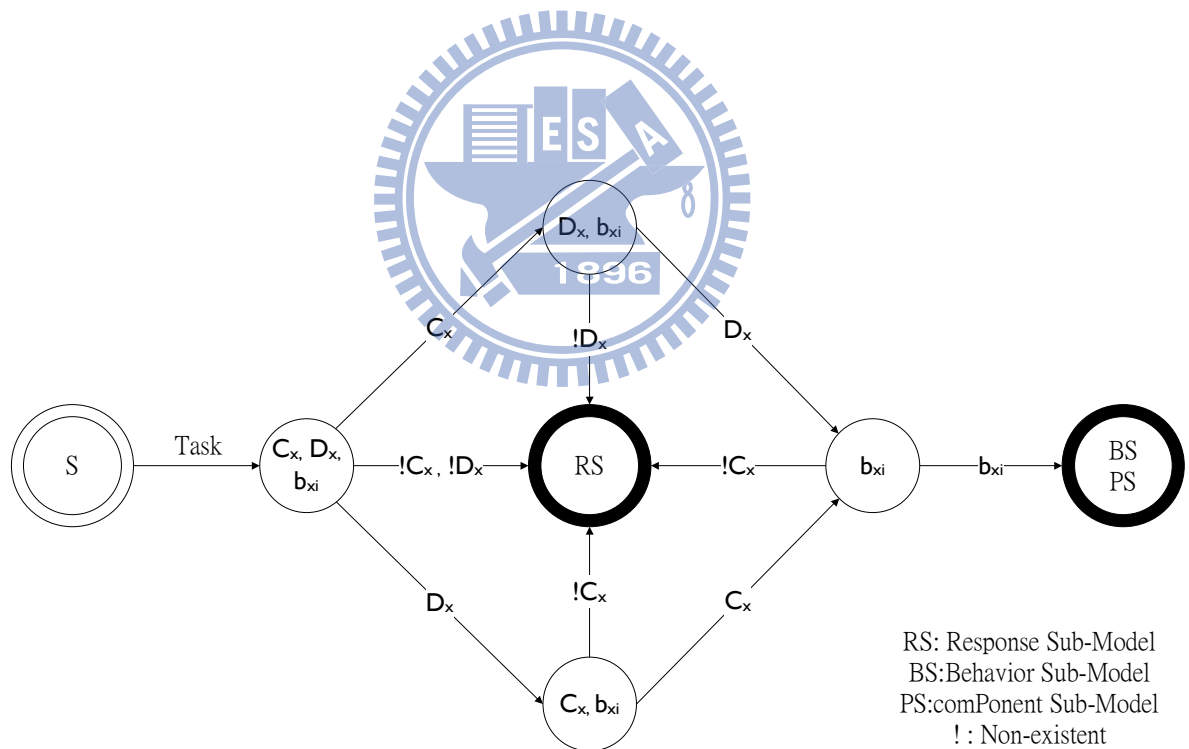


Figure 3.3: Major Flow

Overview As mentioned component, there is a overview of the dialogue flow that composed by those components, as Figure. 3.4. At beginning, the Natural Language ToolKit decomposes the sentence to several parts and give the words simple tags. And the task generator stores the useful words by the mentioned tags by some string manipulation. The component sub-model, the behavior sub-model and the missing information sub-model process respective task synchronously. The component sub-model and the behavior sub-model find the corresponding command form the mapping tables. Updating the mapping table is for the behavior sub-model and the task generator. The missing information sub-model add missing information in command file according to user profile preference.

Task Generator A flow of task generator is Figure. 3.5 Then there is the task generator inspects whether a new device C_x in user command. The purpose of task generator is to generate a task. At first, task generator checks he device C_x whether in dataset. If there is a new device, the task generator will register the device in dataset. There is a logical rule to add new device. For example, if the last device type id is 1000 and the new device type will be 1001.

Another function of task generator is checking if the behavior b_{xi} and D_x is in user command. But the task generator does not update the behavior mapping table.

Component Sub-model There is a flow of the component sub-model as show in Figure. 3.6. The component sub-model checks whether the device C_x and the description of the device D_x is in user command. If there is no device C_x , component sub-model will ignore the device type column in command file. And then, the component sub-model examines the description D_x weather is in the dataset. If there are new descriptions, the component sub-model registers the new description in dataset. There still is a logical rule to add new description of device. If

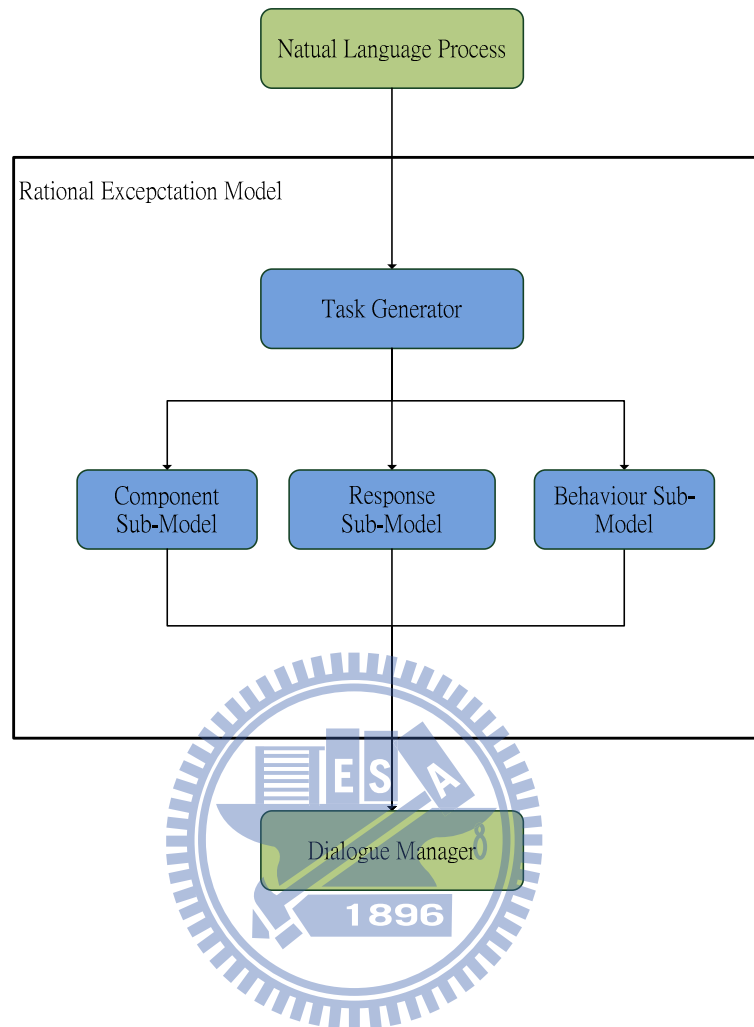


Figure 3.4: Overview

there is no description of device D_x , the component sub-model still ignores the device description column in command file.

Pseudo Code

```
Step 1 : input  $C_x$  and  $D_x$ 
Step 2 : if  $C_x$  exists in component mapping table
            $C_x \rightarrow$  command file
         else
           break
Step 3 : if  $D_x$  exist in component mapping table then
           if device column =  $C_x$ 
              $D_x \rightarrow$  command file
           else  $C_x \rightarrow$  device column
         else
            $D_x \rightarrow$  component mapping table
```

Behavior Sub-model There is a flow of the behavior sub-model as show in Figure. 3.7. At first, the behavior sub-model still checks the behavior of device b_{xi} whether is in dataset. If there is a new behavior, the behavior sub-model registers the new behavior of device in dataset. And the behavior sub-model sets the device type column in the behavior b_{xi} and fills the behavior b_{xi} in command file.

On the other hand, if there is an old behavior, the behavior sub-model must check the flag F_{NewC} . If the flag is 1, the behavior sub-model would tag the behavior b_{xi} to the corresponding device C_x . Behaviour sub-model judges whether the behavior from user command is logical for device C_x . If the behavior b_{xi} is logical, behavior sub-model fills b_{xi} in command file according to behavior mapping table. The behaviour sub-model ignores the b_{xi} in user command, if the behavior b_{xi} is not logical.

Pseudo Code

```
Step 1 : input  $b_x$  and  $C_x$ 
Step 2 : if  $b_x$  exists in behavior mapping table
    if device column = NULL
         $C_x \rightarrow$  device column
    if device column =  $C_x$ 
         $b_x \rightarrow$  command file
    else
        break
else
     $b_x \rightarrow$  behavior mapping table
     $C_x \rightarrow$  behavior mapping table
```

Missing Information Sub-model There is a flow of the missing information sub-model as show in Figure. 3.8. At first, the missing information sub-model checks the user command. If there is not device C_x in user command, the missing information sub-model fills device type ID in command file according to type ID column behavior b_{xi} from behavior mapping table. Then missing information sub-model examines the location in user command. If there is no location information in user command, the missing information sub-model gets location information from user profile preference. And the missing information sub-model also examines the feature information in user command, as adjective, brand information or other information describes the device C_x . But the feature information is not necessary.

Pseudo Code

```
Step 1 : input  $b_x$ 
Step 2 :  $C_x =$  device column of  $b_x$ 
Step 3 :  $C_x \rightarrow$  command file
Step 4 :  $D_x =$  description of  $C_x$ 
Step 5 :  $D_x \rightarrow$  command file
```

3.5 Infer missing information

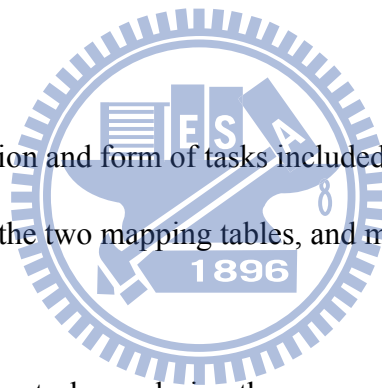
The mapping table would be solution for how to infer the missing information. By the relationship between C_x and b_x , the behavior sub-model would infer the corresponding behaviors for the device. And so does the missing information sub-model.

The sub-models with inferring function still deduce the wrong solution, because the faulty mapping table. But by the weight column, the sub-models would refine the mapping table. The probability of inferring the wrong solution would decrease, by refining the weight over and over.

3.6 Summary

We introduce the definition and form of tasks included notations to express users' commands. We still defined the two mapping tables, and make a description of the sub-models in function and flowchart.

For different elements in a task, we design the corresponding sub-models to deal with different sceneries. The sub-models would check the elements in a task and missing information, and do the actions for different sceneries.



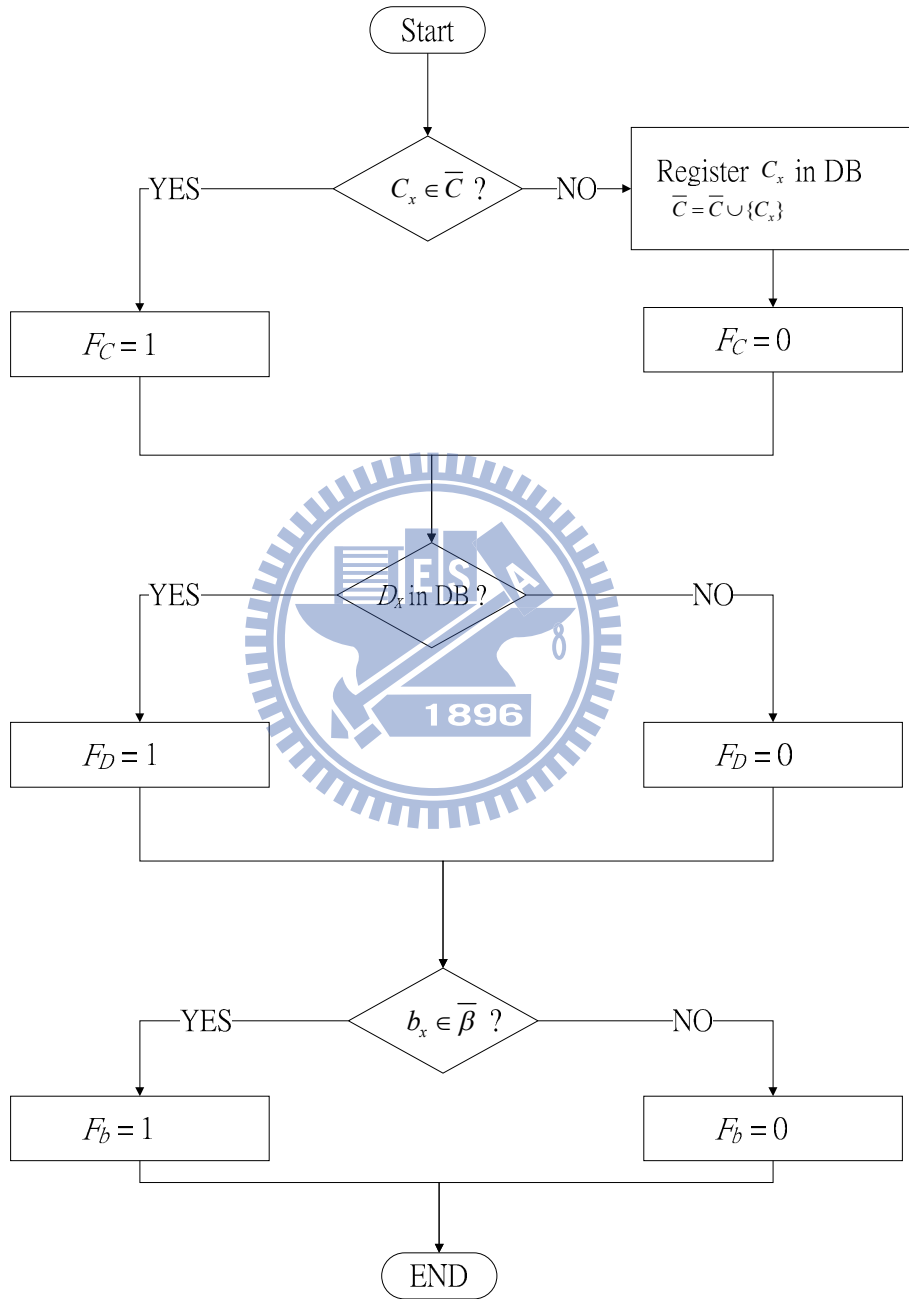


Figure 3.5: Task Generator

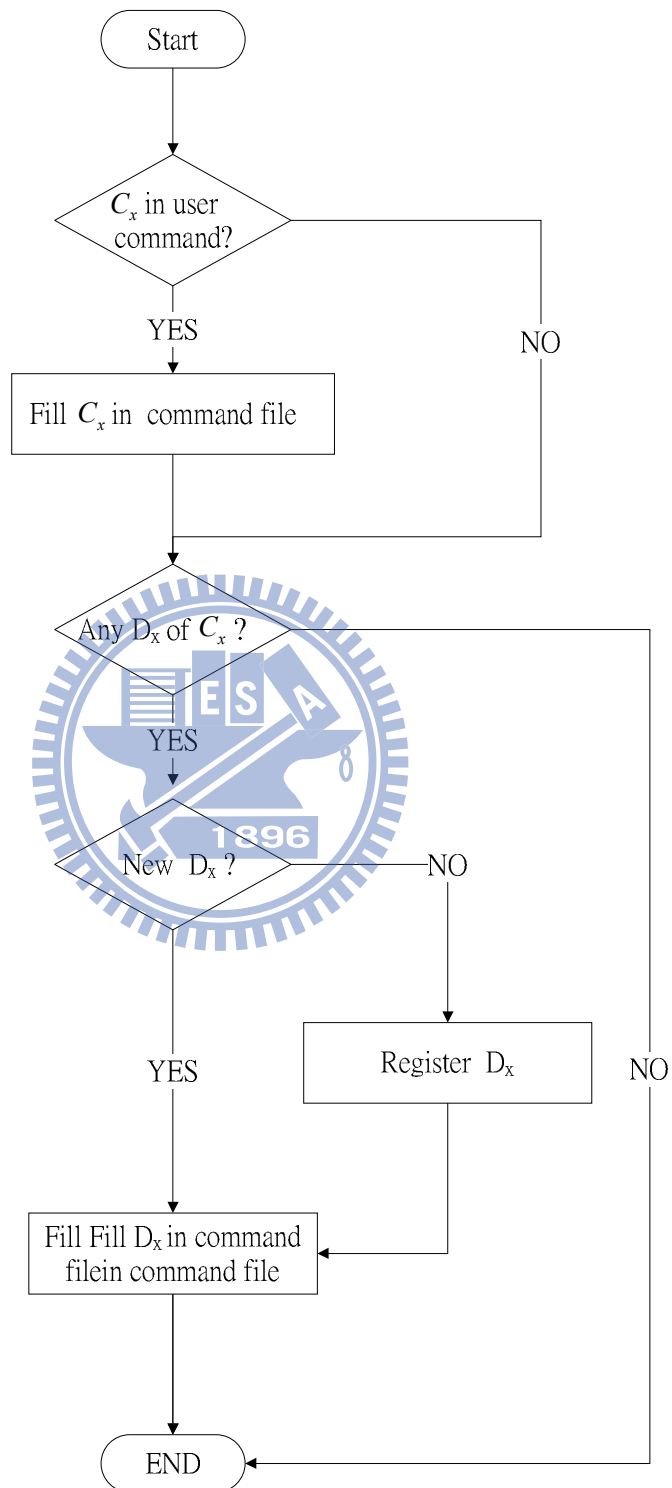


Figure 3.6: Component Manager

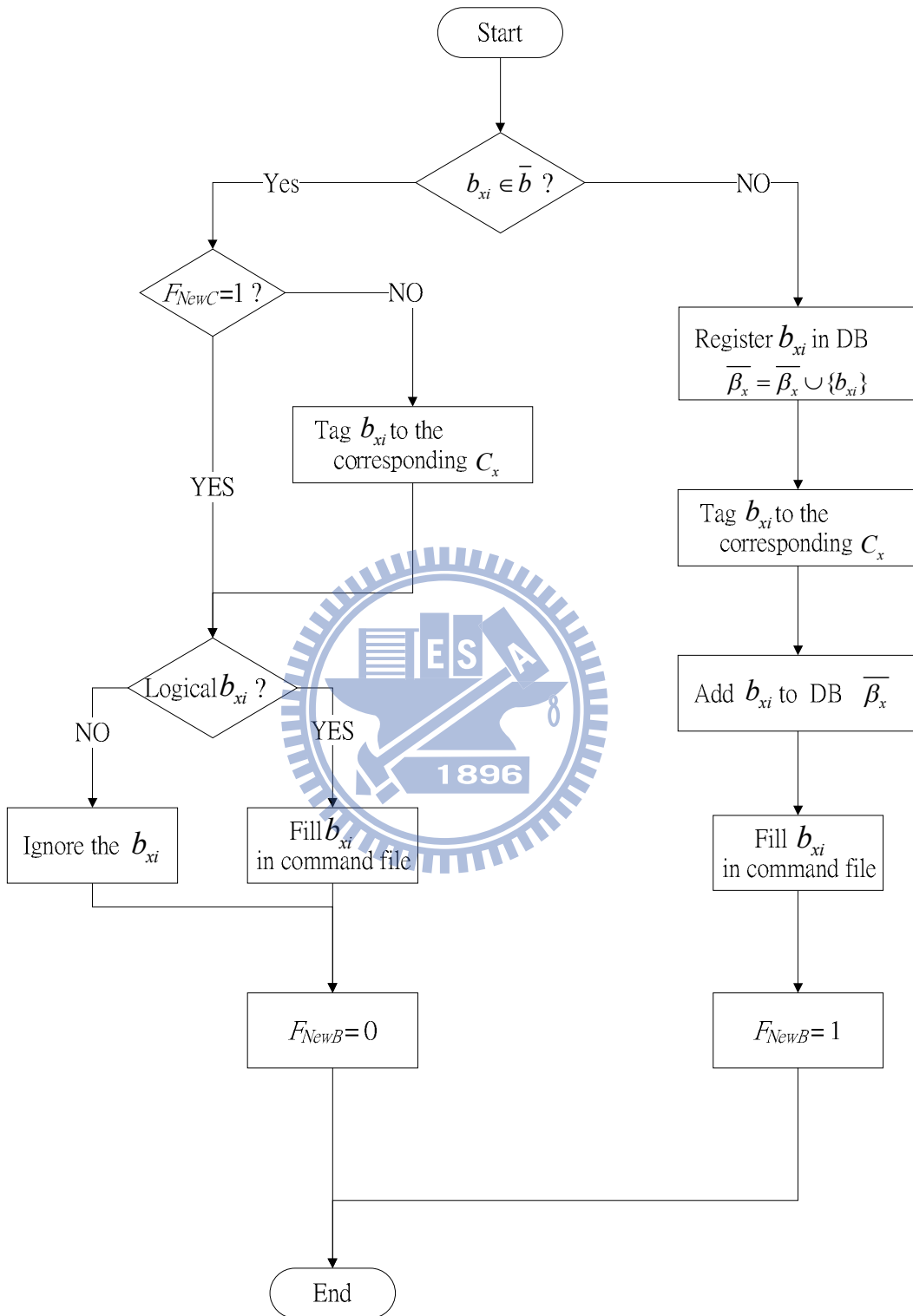


Figure 3.7: Behavior Manager

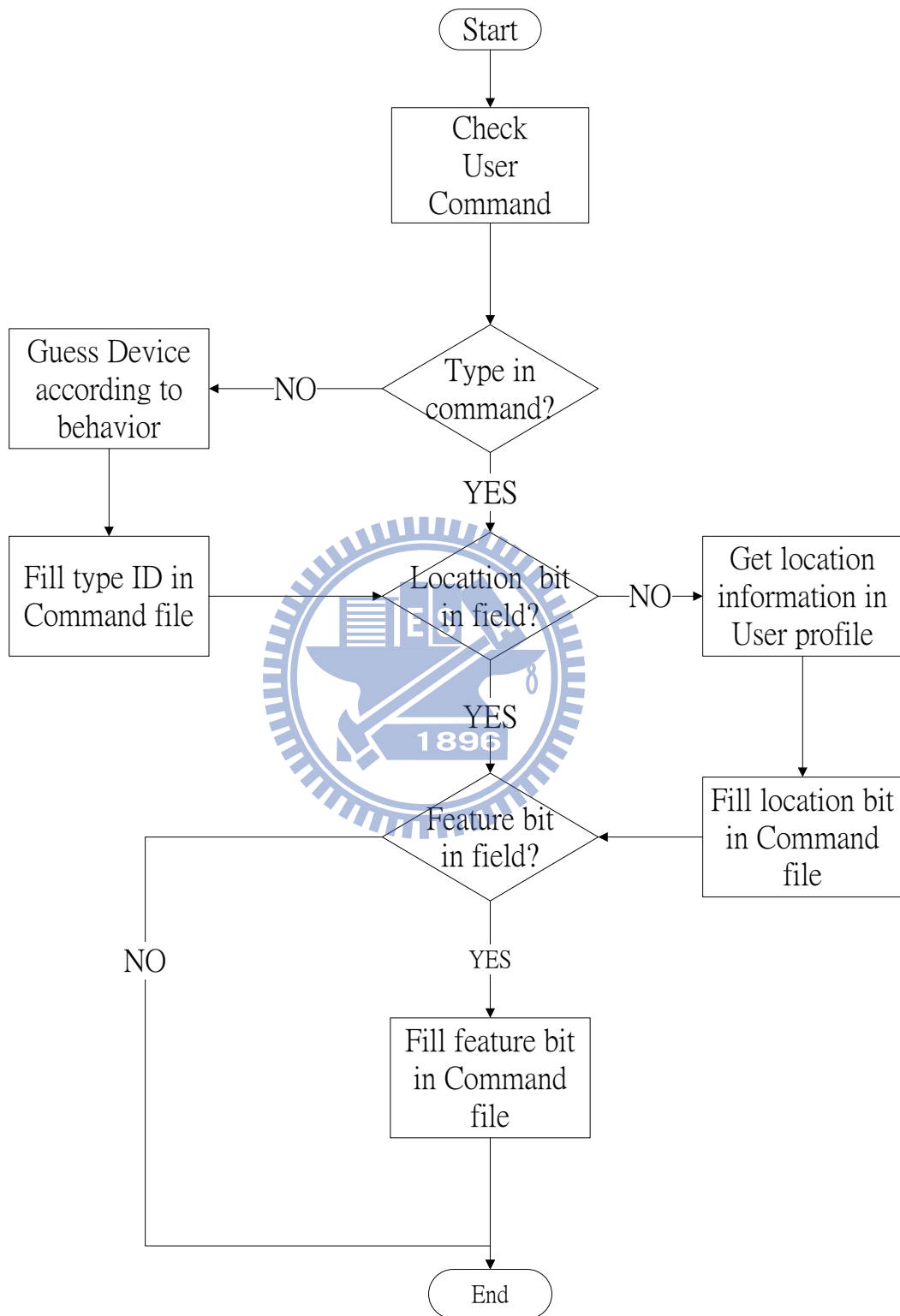


Figure 3.8: Response Manager

Chapter 4

Implementation

In the chapter we will discuss our implementations of REM based on our proposed model.

4.1 Environment

We divide into two parts to explain how we implement. One is hardware, the other is software included the tools we applied to.

4.1.1 Hardware

In hardware, the CPU is **Intel C2D E7500**, it's a dual-core CPU. The RAM is 2048 MB. And we use keyboard as input, and the output is monitor. In fact, a single-core CPU still satisfies the demand of our model.

There is a list about the hardware. (Table 4.1)

4.1.2 Software

For general environment, all tools we applied work normally on Windows XP. We have to simulate Windows XP with virtual machine. So we use VMware 3.0.1 to construct Windows

Table 4.1: Hardware List

CPU	Intel C2D E7500
RAM	2048MB
Input	Keyboard
output	Monitor

Table 4.2:

Software	Version
VMware	3.0.1
Eclipse	helios-SR2
OAA	2.7
GCC	N/A

XP. And the host operated system is Windows 7

The program editor is **eclipse** and its version is **Helios Service Release 2**. But there is no C/C++ compiler in **eclipse**. Before writing the program, we must install the compiler **gcc**.

At first, the program language of Natural Language ToolKit (NLTK) with distributions for Windows, Mac OSX and Linux is **Python**. So, we have to write a simple program to apply NLTK. The version of **Python** is 2.6. To use the NLTK library, we need to import the head file. And the version of OAA library is 2.3.2 since June, 2007.

The main program language we applied to complete all components is C++.

4.2 Open Agent Architecture

We would introduce the OAA in this section. We have to introduce the concept of agents at first. And then we would dig in OAA.

4.2.1 Agent-Based Model

Agent-based model is based on Artificial Intelligence (AI) and focus on dialogue system as cooperation between intelligent agents. All of these approaches for agents do focus on "Goal", "Solution" and "Event". "Event" means the user's request, "Goal" is the expected action for user's request and "Solution" is the actual action for user's request.

When the agents what are build-in dialogue system receive "Event", the agents would find

"Solution" and send "Goal" to another agent or user. There are not only one "Solution" for a "Event" , and the "Solution" will change according to some parameters.

In agent-based model shows as Figure. 4.1, a small black point represents a agent with some function. The gray ellipse is the external environment, and the other ellipses in the big gray ellipse are perceived by agents. A agent is responsible for perceiving a small part of environment, that means function of agents is not strong. If there are two agents, one of them is in charge of handling what kind of home appliance user operates and the other one is responsible for searching which one to be operated.

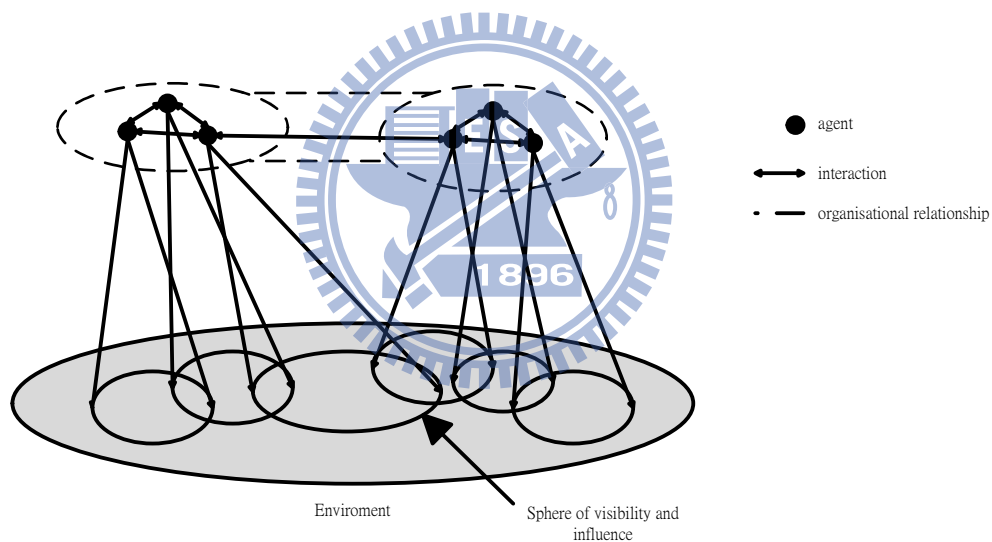


Figure 4.1: Agent-Based Model

4.2.2 Structure of OAA

The OAA is shown as Figure. 4.2. The main component in OAA is **facilitator**, being responsible for distributing the task to the specified agent. All agents must register their solutions in facilitator, or the facilitator would not pass the task the the agent without

registering. The Interagent Communication Language (ICL) developed by SRI is the form of communication between agents and the facilitator.

In 1996, the Agent Development Toolkit (ADT) [21] is proposed as a IDE. And the OAA is wide range to be applied. As Section 2 mentioned, the MIMUS applied the OAA in its architecture.

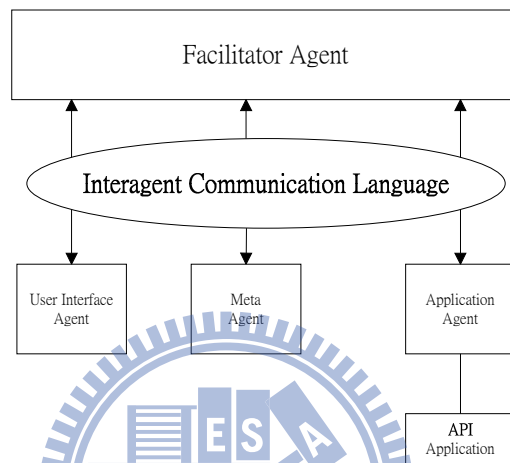


Figure 4.2: Open Agent Architecture

4.3 Structure of Proposed Mode with OAA

We introduce the structure of our model with the hybrid approach, as shown Figure 4.3. As mentioned in Section 3, the NLP would analysis the sentence from the user. The task generator extracts the tagged word from the result form the NLP. The facilitator would pass the tagged word to the appropriate manager.

At first, we have to launch the facilitator, and we launch the component manager and the behavior manager, as shown Figure 4.4 The behavior manager registers its function in facilitator.

There are two managers invoked in runtime. as shown Figure 4.5. Obviously, the

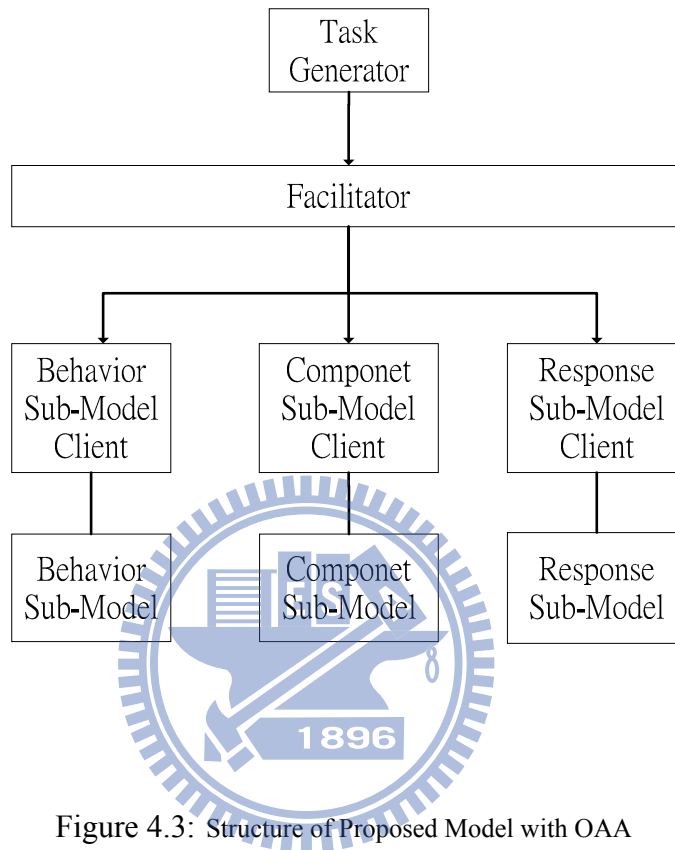


Figure 4.3: Structure of Proposed Model with OAA

facilitator accepts the requests of registering from the two managers.

4.4 summary

In summary, we introduce the hardware and software environment in our design We introduce the architecture of Open Agent Architecture (OAA),and how we use the structure of OAA to design and implement REM in mentioned environment.

```

命令提示字元 - fac.exe
Facilitator version info: v. 2.3.2
Compiled with:
  OAA library: v. [2,3,2]
  Compound queries: NO
  Backward compatibility mode: YES
-----
Listening at tcp(gogogohu-f0aca3,3378)

addr(tcp(192.168.1.241,3378)) <root> can solve:
  [solvable(agent_data<_31,_32,_33,_34,_35,_36>,[type(data)]),solvable(agent_host<_52,_53,_54>,[type(data)]),agent_version<_66,_67,_68>,[type(data)]),solvable(facilitator_data<_76,_77,_78,_79,_80>,[type(data)]),can_solve<_96,_97>,[type(data)]),solvable(agent_location<_105,_106,_107,_108>,[type(data)]),solvable(data<_124,_125>,[type(data)]),solvable(icl_type<_141,_142>,[type(data)])]

Ready.

2011/6/29 5:24:35
Accepting handshake from 2 <Compoente_manager>
addr(tcp(192.168.1.241,3378),2) <Compoente_manager> can solve:
  [solvable(component_search<_6224,_6258,_6290,_6320>,[callback(component_search)])]

```

Figure 4.4: Facilitator with One Manager



```

命令提示字元 - fac.exe
Listening at tcp(gogogohu-f0aca3,3378)

addr(tcp(192.168.1.241,3378)) <root> can solve:
  [solvable(agent_data<_31,_32,_33,_34,_35,_36>,[type(data)]),solvable(agent_host<_52,_53,_54>,[type(data)]),agent_version<_66,_67,_68>,[type(data)]),solvable(facilitator_data<_76,_77,_78,_79,_80>,[type(data)]),can_solve<_96,_97>,[type(data)]),solvable(agent_location<_105,_106,_107,_108>,[type(data)]),solvable(data<_124,_125>,[type(data)]),solvable(icl_type<_141,_142>,[type(data)])]

Ready.

2011/6/29 5:24:35
Accepting handshake from 2 <Compoente_manager>
addr(tcp(192.168.1.241,3378),2) <Compoente_manager> can solve:
  [solvable(component_search<_6224,_6258,_6290,_6320>,[callback(component_search)])]

2011/6/29 5:26:11
Accepting handshake from 4 <Behavoir_manager>
addr(tcp(192.168.1.241,3378),4) <Behavoir_manager> can solve:
  [solvable(behavior<_6246,_6276,_6306>,[callback(behavior)])]

```

Figure 4.5: Facilitator with Two Managers

Chapter 5

Analysis

In this chapter, we would introduce the actions of REM in four case. In Section 5.2, we discuss the effect of REM in the dialogue manager with different approach.

5.1 Case Study

According the task we defined before, we list several situations about the missing information of the task. we would continue using the notation C_x , b_x and D_x to composed a task.

We denote the verbs to b_{xi} and nouns to C_x based on our model. We list all possible cases.

I. Task with C_x

II. Task without C_x

III. Task with b_x

IV. Task without b_x

There is a table as show (Table 5.1)

We follow the definition of degree of complexity allocated to each task.[17] A simple task includes only one b_{xi} , a complex task requires more than one b_{xi} . On the other hand, the task what remembered the previous information is complex, as referential relations.[22][23]

We analysis four cases. In our proposed model, we can handle more than one b_{xi} . In following case, we just discuss the situation with one b_{xi} .

In fact, we can break a task with more than one b_{xi} into several sub-task with one b_{xi} . On

Table 5.1: Case Study

		NP	
		I	II
VP	III	Case 1	Case 2
	IV	Case 3	Case 4

the other side, the command sending to devices is depended on b_{xi} not previous device information. So, there is no need to analysis overly complex tasks in this paper.

The situation of lack of D_x is similar with lack of C_x . So, we only discuss the lack of C_x .

5.1.1 Case 1

In case 1, our system receives the sentence with C_x and b_{xi} . In principles, the actions of system are independent because the independence of behaviour manager and component manager. So there is a new b_{xi} , the system registers the b_{xi} in dataset and update the device ID of b_{xi} . And there is a new C_x , the system registers the C_x in dataset and update the device ID of b_{xi} .

As mentioned above, we analysis four situation as show (Table 5.2)

Existing b_{xi} and Existing C_x It's a general situation with complete sentence. All b_{xi} and C_x in user command existed in dataset. Because of this, our system acts normally.

Existing b_{xi} and new C_x If there is only C_x not in the dataset, our system still registers the C_x and update device ID in existed b_{xi} . Next, our system updates the device ID of b_{xi} . In doing so, our system can judge the logic of b_{xi} next time.

New b_{xi} and Existing C_x If there is only b_{xi} not in the dataset, our system still registers the b_{xi} and update device ID in b_{xi} . But, no need to register the C_x . The situation is very similar to the previous.

Table 5.2: Case 1

		I C_x in NPs	
		Existing C_x	New C_x
III b_{xi} in VPs	Existing b_{xi}	No register No update	Register C_x Update device ID of b_{xi}
	New b_{xi}	Register b_{xi} Update device ID of b_{xi}	Register C_x and b_{xi} Register device ID in b_{xi}

Table 5.3: Example of Case 1

Turn on the TV in the bedroom	
[Turn on] is an existing b_{xi} , [TV] is an existing C_x	No register No update
[Turn on] is an new b_{xi} , [TV] is an existing C_x	Register [Turn on] Update device ID in [Turn on]
[Turn on] is an existing b_{xi} , [TV] is an new C_x	Register [TV] Update device ID in [Turn on]
[Turn on] is an new b_{xi} , [TV] is an new C_x	Register [Turn on] and [TV] Update device ID in [Turn on]

New b_{xi} and new C_x Even the sentence includes b_{xi} and C_x , the b_{xi} and C_x are not in dataset.

Our system not only registers b_{xi} and C_x in dataset but also update the device ID in b_{xi} . In doing so, our system can distinguish the b_{xi} and C_x next time.

We take a t for a example. 5.3

5.1.2 Case 2

In this case, the dominated factor is the C_x . The factor C_x would dominate the action of our system. If there is a new C_x , the system have to register the C_x . On the other hand, the system loads the b_{xi} when the C_x is existed in dataset. In fact, We define a component not only device C_x but also description of device D_x , including location and features. We use C_x and D_x to describe a complete component.

The point of case 2 is system how to handle the situation that there is not any b_{xi} in user command but C_x . All decision of case 2 are up to C_x . We take a sentence for a example. 5.5

Table 5.4: Case 2

		II C_x in NPs	
		Existing C_x	New C_x
III b_{xi} in VPs	NULL	Load b_{xi} according to user file	Register C_x
	NULL	Load b_{xi} according to user file	Register C_x

Table 5.5: Example of Case 2

TV in the bedroom	
[TV] is an existing C_x	Load b_{xi} according to user preference
[TV] is a new C_x	Register [TV]

And there is a table as show (Table 5.4)

5.1.3 Case 3

In this case, we pay attention on b_{xi} -- lack of device type, only behaviour, as show (Table 5.6). The user's input includes only behaviour b_{xi} . we can expect the b_{xi} dominates the flow of our system.

There are two possible action. First, user gives a b_{xi} which the system can't realize. Our system registers the b_{xi} without device ID, because there's no C_x in user's input. Second, b_{xi} form user's sentence exists in dataset. Our system can load the existed C_x from device ID of b_{xi} according to the behaviour mapping table.

We take a sentence for a example. 5.7

Table 5.6: Case 3

		I C_x in NPs	
		NULL	NULL
III b_{xi} in VPs	Existing b_{xi}	Load device ID from b_{xi}	Load device ID from b_{xi}
	New b_{xi}	Register b_{xi} , but no device ID	Register b_{xi} , but no device ID

Table 5.7: Example of Case 3

Turn it on in the bedroom	
[Turn on] is an existing b_{xi}	Load device ID from [Turn on]
[Turn on] is a new b_{xi}	Register [Turn on]

5.1.4 Case 4

There is a situation we do not handle. A sentence without any b_{xi} and C_x is not a legal command. Because lack of information can not be a complete command, we preclude this situation.

5.2 Discussion

We would discuss the dialogue systems with REM or without REM. As we mentioned Section ??, the REM would fill the missing information in a task. The dialogue manager with REM would receive the task with C_x , b_x and D_x , if REM could find the solution. The dialogue manager without REM would query database or ask user for missing information.

5.2.1 Finite-state approach with REM

The dialogue manager in finite-state approach needs the a series of ordered words or phrases to be input. The dialogue manager with FSM should check the elements in task in ordered. If the input is not intact, the dialogue manager may stuck.

The REM would deal with the missing information, the dialogue manager could receive the task with full information. The dialogue manager with REM could decrease the size of dialogue and work well.

5.2.2 Frame-based approach with REM

The dialogue manager with frame-based approach fills the designed slots. The element in a task which we determined would be filled in the designed slots. The dialogue manager with frame-based approach would ask users for missing information.

The sub-models in REM could guess the missing information. The dialogue manager could receive the result from REM to fill the slots, not to ask users for missing information.



Chapter 6

Conclusion

The REM could decrease the complex process querying data of dialogue manager about missing information. The dialogue manager does not ask user after the REM.

The basic concept of REM is finite state machine with a series of judging. Because the form of the task is fixed, we apply the finite state machine in determine mechanism even the finite state machine with shortage in scalability. The sub-models deal with the missing information and the unknown element in a task.

If missing information happens, the REM works based on the information of task, as device or behavior. The REM would choose different action based on the device or behavior in the task. With behavior mapping table or component mapping table, the REM could find the corresponding solution to the task with missing information.

By the determined task, we design the REM with distributed agents for missing information. The form of a task would assist REM to diagnose the state of a task, sub-models in REM could find the solution.

The Rational Exception Model (REM) would help the dialogue manager to analysis the state of dialogue. The sub-models in REM would query the missing information and determine the logic of behavior for corresponding device with distributed agents. The dialogue manager does not query the missing information with mentioned approached.

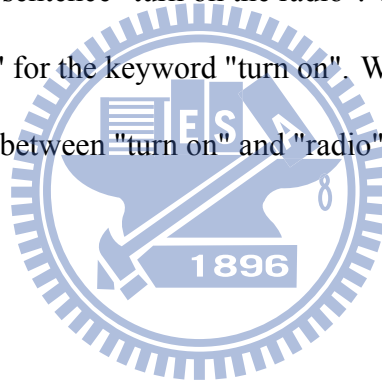
Chapter 7

Future Work

Future work in our research is to refine the mapping table with more efficient approach.

We choose the simple approach, weight column in mapping table, to infer the solution.

For this purpose, we could query database on-line and determine the relationship between the keyword we searched and the result. For example, we could search the keyword "turn on", the result would be the short sentence "turn on the radio". The short sentence included the corresponding device "radio" for the keyword "turn on". We can refine our mapping table by determining the relationship between "turn on" and "radio".



References

- [1] D. Traum and S. Larsson, "The information state approach to dialogue management," in *Current and New Directions in Discourse and Dialogue*, 2003, pp. 325--353.
- [2] A. Cheyer and D. Martin, "The open agent architecture," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 4, no. 1, pp. 143--148, March 2001, oAA.
- [3] M. F. McTear, "Spoken dialogue technology: enabling the conversational user interface," *ACM Comput. Surv.*, vol. 34, pp. 90--169, March 2002. [Online]. Available: <http://doi.acm.org/10.1145/505282.505285>
- [4] M. M. University and M. F. Mctear, "Modelling spoken dialogues with state transition diagrams: experiences with the cslu toolkit," in *Proc 5th International Conference on Spoken Language Processing*, 1998, pp. 1223--1226.
- [5] B. Hansen, D. G. Novick, and S. Sutton, "Systematic design of spoken prompts," in *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, ser. CHI '96. New York, NY, USA: ACM, 1996, pp. 157--164. [Online]. Available: <http://doi.acm.org/10.1145/238386.238466>
- [6] H. Aust, M. Oerder, F. Seide, and V. Steinbiss, "The philips automatic train timetable information system," *Speech Commun.*, vol. 17, pp. 249--262, November 1995. [Online]. Available: <http://portal.acm.org/citation.cfm?id=219030.219079>
- [7] D. S.-H. C. Chin-Han Tsai, "A study on speech dialogue system and dialogue strategy," July 2005.

- [8] M. Guo, Y. Liu, and J. Malec, "A new q-learning algorithm based on the metropolis criterion," *EEE Trans Syst Man Cybern B Cybern*, vol. 34, pp. 2140--3, 2004. [Online]. Available:
<http://www.biomedsearch.com/nih/new-Q-learning-algorithm-based/15503510.html>
- [9] E. Levin, R. Pieraccini, and W. Eckert, "Using markov decision process for learning dialogue strategies," in *Proc. ICASSP*, 1998, pp. 201--204.
- [10] H. machine-dialog Corpora, W. Eckert, E. N \square th, H. Niemann, and E.-G. Schukat-Talamazzini, "Real users behave weird - experiences made collecting large human-machine-dialog corpora," 1995.
- [11] S. Sutton, R. Cole, J. D. Villiers, J. Schalkwyk, P. Vermeulen, M. Macon, Y. Yan, E. Kaiser, B. Rundle, K. Shobaki, P. Hosom, A. Kain, Johan, J. Wouters, D. Massaro, and M. Cohen, "Universal speech tools: The cslu toolkit," in *In Proceedings of the International Conference on Spoken Language Processing (ICSLP*, 1998, pp. 3221--3224.
- [12] M. F. Mctear, "Using the cslu toolkit for practicals in spoken dialogue technology," in *University College London*, 1999, pp. 1--7.
- [13] C. L. Liu, *Elements of discrete mathematics*, 1977.
- [14] J. Chu-Carroll, "Mimic: an adaptive mixed initiative spoken dialogue system for information queries," in *Proceedings of the sixth conference on Applied natural language processing*, ser. ANLC '00. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000, pp. 97--104. [Online]. Available:
<http://dx.doi.org/10.3115/974147.974161>

- [15] J. Chu-Carroll and M. K. Brown, "An evidential model for tracking initiative in collaborative dialogue interactions," *User Modeling and User-Adapted Interaction*, vol. 8, pp. 215--254, February 1998. [Online]. Available: <http://portal.acm.org/citation.cfm?id=598279.598319>
- [16] J. G. Amores, G. Pérez, and P. Manchón, "Mimus: a multimodal and multilingual dialogue system for the home domain," in *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ser. ACL '07. Morristown, NJ, USA: Association for Computational Linguistics, 2007, pp. 1--4. [Online]. Available: <http://portal.acm.org/citation.cfm?id=1557769.1557771>
- [17] P. Manchón, C. del Solar, G. Amores, and G. Pérez, "Multimodal interaction analysis in a smart house," in *Proceedings of the 9th international conference on Multimodal interfaces*, ser. ICMI '07. New York, NY, USA: ACM, 2007, pp. 327--334. [Online]. Available: <http://doi.acm.org/10.1145/1322192.1322249>
- [18] T. Project., *Talk and Look: Linguistic Tools for Ambient Linguistic Knowledg*, 2007. [Online]. Available: www.talk-project.org
- [19] P. M. Portillo, G. P. García, and G. A. Carredano, "Multimodal fusion: a new hybrid strategy for dialogue systems," in *Proceedings of the 8th international conference on Multimodal interfaces*, ser. ICMI '06. New York, NY, USA: ACM, 2006, pp. 357--363. [Online]. Available: <http://doi.acm.org/10.1145/1180995.1181061>
- [20] N. Chomsky, *Syntactic structures*, 1957.
- [21] A. C. G.-L. L. David L. Martin, "Development tools for the open agent architecture," vol. PAAM 96. SRI AI center, April 1996.

- [22] L. Ahrenberg, A. Jönsson, and N. Dahlbäck, "Discourse representation and discourse management for a natural language dialogue system," in *In Proceedings of the Second Nordic Conference on Text Comprehension in Man and Machine*, Taby, 1990.
- [23] J. Hawkins, "Definiteness and indefiniteness: A study in reference and grammaticality prediction," 1978.

