Universität Karlsruhe (TH)

Using POMDP for Person Identification Dialogs

Studienarbeit am Institut Interactive Systems Labs
Prof. Dr. Alex Waibel
Fakultät für Informatik
Universität Karlsruhe (TH)

von

cand. inform.
Paul Märgner

Betreuer:
Prof. Dr. Alex Waibel
Dr. Hartwig Holzapfel

Tag der Anmeldung: 1. Juni 2009
Tag der Abgabe: 31. August 2009

Interactive Systems Labs

Kurzfassung

Das Ziel dieses Projektes ist die Realisierung eines Personen-Identifikations-Dialogs, welcher explizite Konfidenzwerte verwendet. Als Dialogmanager wurde ein partiell beobachtbarer Markow-Entscheidungsprozess (Partially Observable Markov Decision Process, POMDP) gewählt. POMDPs ermöglichen Unsicherheiten im Dialog explizit zu modellieren und sind dadurch potenziell besser für natürlich sprachliche Dialoge geeignet. Weiterhin wurde in vielen Arbeiten gezeigt, dass POMDPs bessere Ergebnisse in Dialogen erzielen als handgeschriebene Dialogstrategien. Tests im Rahmen dieser Arbeit haben ergeben, dass die vorhandenen POMDP-Beschreibungstools nicht ausreichend sind für die Beschreibung eines POMDPs mit expliziten Konfidenzwerten. Aus diesem Grund wurde in dieser Arbeit ein grafisches Software Tool (POMDP-Builder) entwickelt, das die Erstellung eines POMDPs mit expliziten Konfidenzen ermöglicht. Dieser POMDP-Builder bietet die Möglichkeit den notwendigen Beschreibungsaufwand zu reduzieren und durch Benutzerführung Eingabefehler zu vermeiden. Dadurch wird insbesondere die Beschreibung komplexer POMDPs erleichtert. Die POMDPs, die mit dem POMDP-Builder erstellt wurden, können in einem gängigen POMDP Beschreibungsformat gespeichert werden. Dies ermöglicht die Verwendung von vorhandenen POMDP Solvern für das Lösen des POMDPs. Im Zuge dieser Arbeit wurde der POMDP-Builder erfolgreich für die Erstellung von zwei Personen-Identifikations-Dialogen eingesetzt. Die Untersuchung dieser Dialoge hat gezeigt, dass der POMDP-Builder deren Beschreibung deutlich vereinfacht. Weiterhin konnte anhand von Tests mit diesen Dialogen belegt werden, dass die Verwendung von expliziten Konfidenzwerten Vorteile in einem Personen-Identifikations-Dialog bietet.

Contents

1 Introduction

1.1 Motivation

In the past years, it has become possible to create a system which can speak with a human in real time. This has been made possible by improvements in voice recognition and semantic analysis. But dialogs of these systems are often unnatural for humans and need much time to reach their goal. The future goal is to build a humanoid robot which is able to speak to a person like a human. It is necessary to build a good dialog-manager with a good dialog strategy to achieve this goal. The dialog strategy selects a decision which is most likely the fastest way to achieve the current goal. Therefore, a good strategy is necessary to complete a dialog fast and successfully. There are many different ways to build a dialog-manager. For example hand-crafted deterministic rules can be used or a Markov Decision Process (MDP) with reinforcement-learning. A new approach is to use a Partially Observable Markov Decision Process (POMDP). Since POMDPs provide a statistical framework which handles uncertainties, they are potentially better for handling dialogs.

1.2 Scope of the Project

The scope of this project is to build a person identification dialog. A person identification dialog is a dialog in which the machine tries to identify the user. The machine can handle new users and known users. The system also uses confidence information from a multimodal user ID system. The confidence is used to improve the recognition performance of known users. Previous work has shown that MDPs and POMDPs can outperform hand-crafted deterministic rules and, in contrast to MDPs, POMDPs are able to model uncertainties. This makes POMDPs potentially better in handling an uncertain environment like a dialog with a human. Thus, a POMDP has been chosen as dialog-manager of the person identification dialog. Several tests have shown that current POMDP toolkits and description methods are insufficient to describe the person identification dialog. Therefore, this work introduces a new approach which can be used to describe a POMDP for the person identification dialog. The description can be converted into a standard POMDP description format which can be used by standard POMDP solvers. This approach has been implemented as the POMDP-Builder which is a graphical software toolkit to define POMDP models which include explicit confidence values. This work has been tested by building a person identification dialog.

1.3 Overview

Chapter 2 gives an overview of the fundamentals important to build a dialog system.

Chapter 3 presents related work briefly.

Chapter 4 describes the POMDP-Builder and the person identification dialog with explicit confidence values.

Chapter 5 describes the building and testing of two example dialogs.

Chapter 6 presents a conclusion.

2 Fundamentals

2.1 Dialog Systems

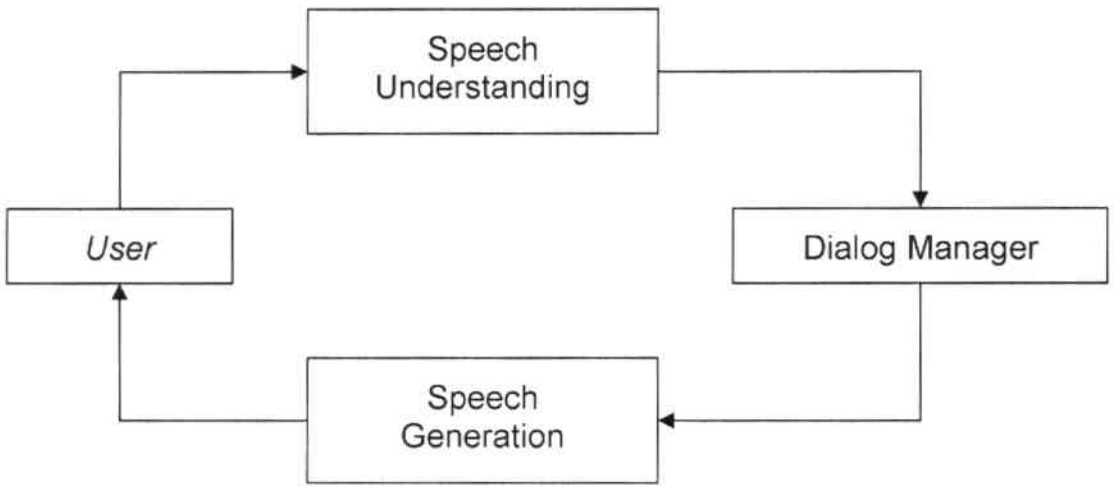


Figure 2.1: Dialog System

The main components of a dialog system are outlined in fig. 2.1. It contains three major components: speech understanding, speech generation and dialog-manager. The speech understanding component maps the user's speech into an abstract user action a_u. This user action contains the semantic information of the user's utterance. The speech generation component does the opposite operation. It gets an abstract machine action a_m and creates the matching speech. The dialog-manager is concerned with decision-making. It decides which machine action a_m is the best in the current state given the current user action a_u. There are different ways to implement a dialog-manager. The following section will introduce three ways to implement a dialog-manager.

2.2 Dialog-Manager

2.2.1 Hand-crafted Deterministic Rules

The conventional way to build a dialog-manager is to create hand-crafted deterministic rules. These rules are used to update the machine-state s_m using the current user action a_u. A dialog policy is needed in addition to these rules. This policy is used to select the next machine action a_m based on the current machine-state. These steps repeat until the goal is satisfied or the dialog fails.

Building this kind of dialog-manager is very time-consuming and the designer has to handle many difficulties. The two major difficulties are uncertainty of the user action and delayed reward. The designer can never be sure that the given user action is correct. Therefore, it is necessary to build a complex system which lets the system recover from this kind of errors. But the main difficulty is the delayed reward. Many decisions made by the dialog-manager do not have an immediate effect but they may have a positive reward in the future. This makes forward planning necessary. But this is very difficult to realize with deterministic rules.

2.2.2 Markov Decision Process (MDP)

One way to solve many of the problems of hand-crafted deterministic rules is to use a statistical approach. A common statistical approach used for dialog systems is a *Markov-Decision-Process* (MDP).

An MDP is defined as a tuple $\{S, A_m, T, R\}$ [Puterman, 1994] [Kaelbling et al., 1995] where

- S is a set of states,
- A_m is a set of machine actions,
- T defines the transition probability $P(s' \mid s, a_m)$. This is the probability that the next state will be $s' \in S$ provided that, in the previous state $s \in S$, the action $a_m \in A_m$ was taken,
- R defines the immediate reward $r(a_m, s)$ received for choosing action a_m in state s.

When an MDP is used as a dialog-manager, the user action is part of the state. With this framework forward planning is possible. The machine can calculate in each state the most likely following state for each action. Then it can calculate the same for these next states and so on. Of course the prediction gets more uncertain with each iteration.

A good way for optimization of the policy is *reinforcement-learning* [Kaelbling et al., 1996]. Reinforcement-learning tries to maximize the reward over the time.

2.2.3 Partially Observable Markov Decision Process (POMDP)

As described in the last section, MDPs provide a good statistical framework to describe a dialog. But it is still assumed the entire state is observable and therefore it is assumed that the entire user is observable. But as mentioned before, the user is not completely observable in most cases in a dialog. This uncertainty is modeled in a *Partially Observable Markov Decision Process* (POMDP). The current state $s \in S$ in a POMDP is always hidden. The machine only estimates how likely the different states are. This estimation over all states is called *belief-state*. The belief-state is often represented by a vector $\vec{b} = (b_1, \ldots, b_n)$ where b_i is the probability that the current state is s_i for all $s_i \in S$. Information about the current state gets the machine from *observations*.

A formal definition of a POMDP according to [Young, 2006] and [Kaelbling et al., 1995] is a tuple $\{S, A, O, T, Z, R, b_0\}$ where

- S is a set of hidden states,
- A_m is a set of machine actions,
- O is a set of observations,
- T defines the transition probability $P(s' \mid s, a_m)$. This is the probability that the next state will be $s' \in S$ provided that, in the previous state $s \in S$, the action $a_m \in A_m$ was taken (*transition model*),
- Z defines the observation probability $P(o' \mid s', a)$. This is the probability that $o' \in O$ is observed in state $s' \in S$ after the machine took action $a \in A$ (*observation model*),
- R defines the immediate reward $r(a_m, s)$ received for choosing action a_m in state s (*reward model*),
- b_0 is the initial belief-state.

The machine selects a machine action $a_m \in A_m$ based on the current belief-state $\vec{b}$. Then the machine receives an observation $o' \in O$. Now the machine calculates the new belief-state $\vec{b}'$ as follows:

$$b'(s') = k \cdot O(o' \mid s', a_m) \cdot \sum_{s \in S} T(s' \mid s, a_m)\, b(s) \tag{2.1}$$

where $k = P\left(o' \mid a_m, \vec{b}\right)^{-1}$ is the normalization factor [Kaelbling et al., 1995].

3 Related Work

3.1 Factored Partially Observable Markov Decision Process

Jason D. Williams, Pascal Poupart and Steve Young introduced a way to represent a dialog-manager as a *factored Partially Observable Markov Decision Process* [Williams et al., 2005]. They took a standard POMDP and separated the POMDP state $s \in S$ into three components: The user's goal $s_u \in S_u$, the user's action $a_u \in A_u$ and the state of the dialog $s_d \in S_d$. Thus, the POMDP state is then represented by a tuple $s = \{s_u, a_u, s_d\}$. They also made the POMDP observation $o \in A_u$. This allows a direct connection between the observed user action o and the real user action a_u in the POMDP state. Note that the state $s = \{s_u, a_u, s_d\}$ is still hidden. They have shown that this factored architecture allows a more detailed specification of a dialog system. But their dialog model was significantly larger than other POMDPs. However, they could show that their factored POMDP has a better performance than several hand-crafted dialog-managers.

3.2 POMDP Toolkit

Trung H. Bui, Boris van Schooten and Dennis Hofs deal with a "Practical dialog-manager development using POMDP" [Bui et al., 2007b]. They addressed several problems of the practical development cycle. They introduced a "POMDP Toolkit" [Bui et al., 2007a]. This toolkit includes a dialog specification parser and an interactive simulator. They created a dialog POMDP specification format (*fpomdp format*) which is based on factored POMDPs [Williams et al., 2005]. The fpomdp format can also contain regular expressions. The parser converts a file in the fpomdp format into a pomdp-file in Tony Cassandra's format [Cassandra, 1999]. Tony Cassandra's format is usually much larger and less readable then the fpomdp format. But Tony Cassandra's format can be loaded by a POMDP solver. Bui et al. used two POMDP solvers, Perseus [Spaan and Vlassis, 2005] and ZMDP [Smith, 2007]. They noticed that ZMDP can handle complex problems much better than Perseus. The interactive simulator provides an easy and fast way to test the policy file created by the solver. It allows acting as the user and observing the machine response.

3.3 Learning and Verification of Names and the Multimodal User ID

Hartwig Holzapfel and Alex Waibel presented a system in which a humanoid robot works as a receptionist [Holzapfel and Waibel, 2008a]. The main focus was on the identification of persons. The robot should be able to recognize known persons and to learn to know new persons. The humanoid robot has a stereo camera for visual perception and distant and close-talk microphones for acoustic perception. These components provide face detection, face identification (face ID), voice identification (voice ID) and speech recognition. The voice and face identification were used to support the recognition of known persons. Furthermore the robot collects the face ID and voice ID from new persons to support the recognition on their next visit. This system was tested with two different dialog-managers: Hand-crafted rules and an MDP trained by reinforcement-learning. The tests have shown that the MDP with reinforcement-learning produces results which are comparable to the results of the hand-crafted system in a known environment. But the MDP with reinforcement-learning has the advantage that it could be automatically retrained for new environments.

In another paper, Hartwig Holzapfel and Alex Waibel introduced an enhanced multimodal user ID [Holzapfel and Waibel, 2008b]. The multimodal user ID combines information which was collected during the dialog using Bayesian networks. This information includes face ID, voice ID and user input like spoken names, spellings and confirmations. Thus, this system is able to combine all information available in the dialog. The tests have shown that the multimodal user ID performs better than a model that relies on dialog information only.

4 Using POMDP for Person Identification Dialogs

4.1 The Idea

The main idea of this work is to build a person identification dialog which includes explicit confidence values. One problem to solve is to choose an appropriate dialog-manager. Many previous works with POMDPs have shown that they show better performance then hand-crafted policies, for example [Williams et al., 2005] and [Bui et al., 2007b]. POMDPs are also able to model uncertainties in the dialog [Young, 2006]. This makes POMDPs the best choice for the dialog-manager. Another important task is the description of the POMDP. Many tests with current tools, like the POMDP Toolkit [Bui et al., 2007a], have shown that they are insufficient to describe a POMDP with explicit confidence values. Therefore, a new approach for the description has been developed in this work. This new approach has been implemented as a graphical software tool (*POMDP-Builder*) which allows the description of a POMDP with explicit confidence values.

4.2 POMDP Builder

The POMDP-Builder is a program with a graphical user interface (GUI) to define a POMDP with explicit confidence values. The most important design decisions are described in the following. The POMDP-Builder uses a GUI because it allows a flexible and interactive input of the data by the designer. The GUI presents the data to the designer in a clear arrangement and the correctness of data during the input process is verified. Both features of a GUI reduce input errors. The POMDP-Builder takes the input data and creates a file in Tony Cassandra's POMDP format [Cassandra, 1999] which can be used as input for current POMDP solvers.

The GUI of the POMDP-Builder consists of six tab-pages. Each of them deals with a different part of the POMDP description. Thus, the designer is guided through the different parts of the description process. Additionally, the input is restricted to valid data only.

In detail, the six parts of the description process are:

1. Definition of states, observations and machine actions.
2. Description of the transition model.
3. Description of the observation model.
4. Description of the reward model.
5. Selection of the initial state.
6. Creating the pomdp-file.

The POMDP-Builder uses a factored architecture where the state is separated into dialog-state, user action and confidence, and the observation is separated into user action and confidence. This factored architecture makes a direct connection between the state and the observation possible. It also allows the designer to use a confidence value in the observation model and also in the transition model. The designer only needs to enter the dialog-states, user actions, confidence-values and machine actions into the first tab-page of the POMDP-Builder GUI (fig. 4.1).

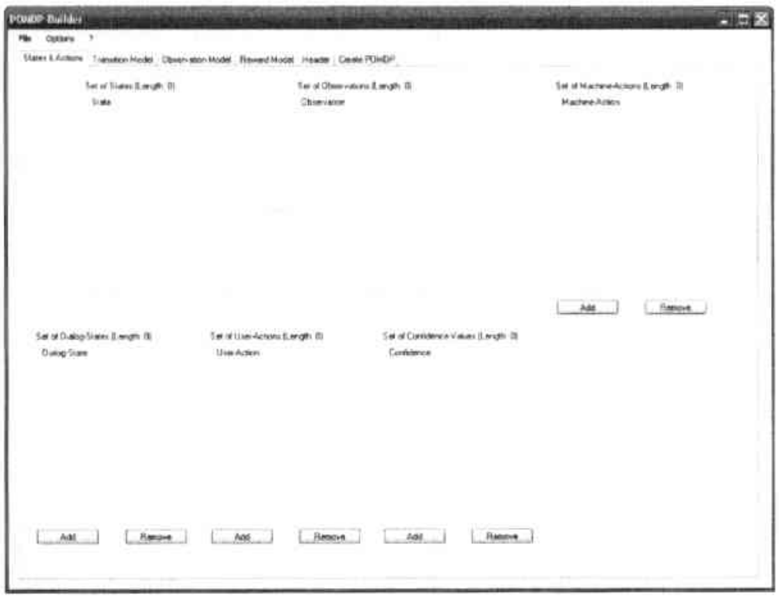

POMDP Builder
File
Options
?
States & Actions
Transition Model
Observation Model
Reward Model
Header
Create POMDP
Set of States (Length: 0)
State
Set of Observations (Length: 0)
Observation
Set of Machine-Actions (Length: 0)
Machine-Action
Set of Dialog-States (Length: 0)
Dialog-State
Set of User-Actions (Length: 0)
User-Action
Set of Confidence-Values (Length: 0)
Confidence
Add
Remove

Figure 4.1: POMDP-Builder: Definition of states, observations and machine actions.

In the next tab-page, the designer can describe the transition model. The transitions are shown in a table on the interface of the POMDP Builder. The transitions in this table can be edited, deleted and rearranged. A new transition probability is added with an interface with several drop-down-lists (fig. 4.3) for Machine-Action, Dialog-State, User-Action, Confidence, next Dialog-State, next User-Action and next Confidence. Each of these drop-down-lists contains the items added on the first tab-page. The designer only needs to select the items. The probability itself is added into a text field. This is the probability that the selected machine action, dialog-state, user action and confidence will lead to next dialog-State, next user-Action and next confidence. This interface is shown in figure 4.2.

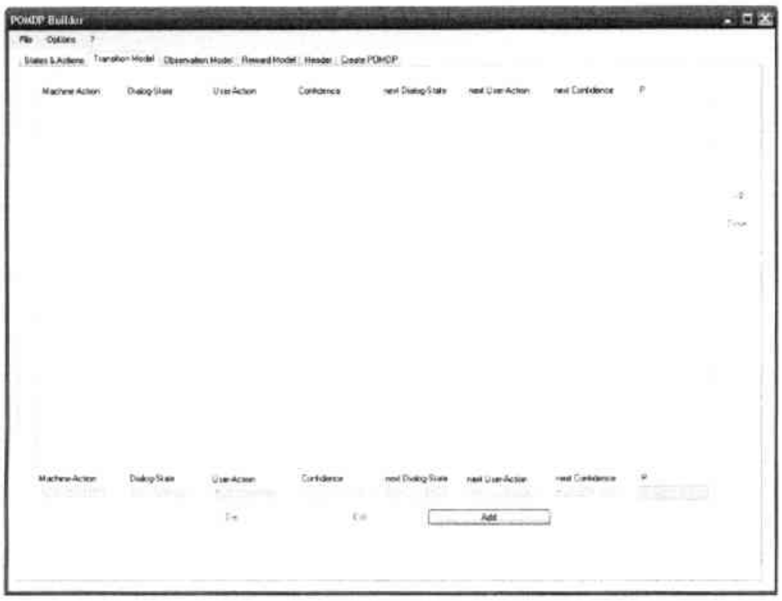


Figure 4.2: POMDP-Builder: Transition model

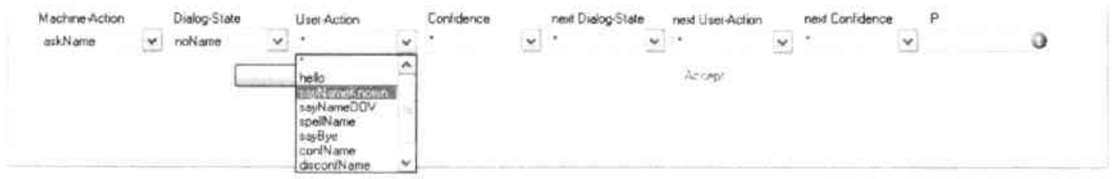


Figure 4.3: POMDP-Builder: Transition model (add)

On the third tab-page, the designer can describe the observation model. There is a table which shows the current observation model. The observation in this table can be edited, deleted and rearranged like in the transition model. The process to add items to the observation model is also very similar to the add-process in the transition model. Again there are several drop-down-lists and a text field for the probability (fig. 4.5). This interface is shown in figure 4.4.

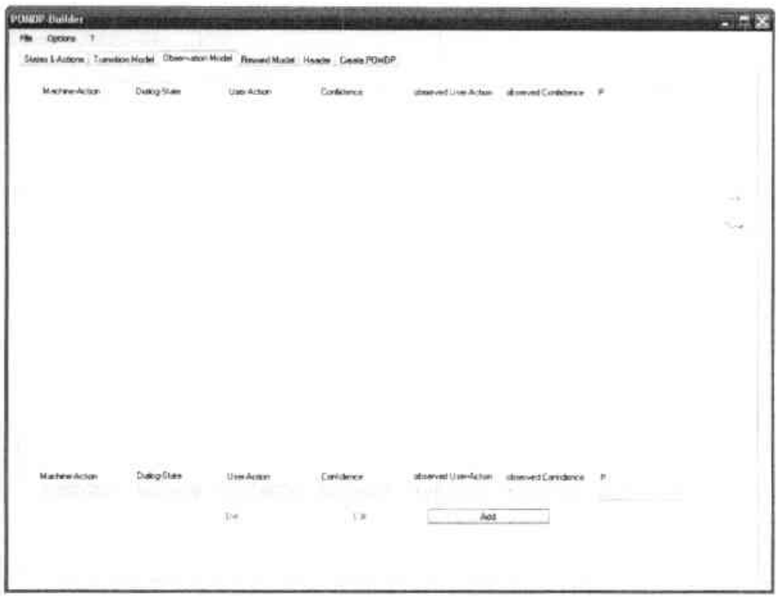
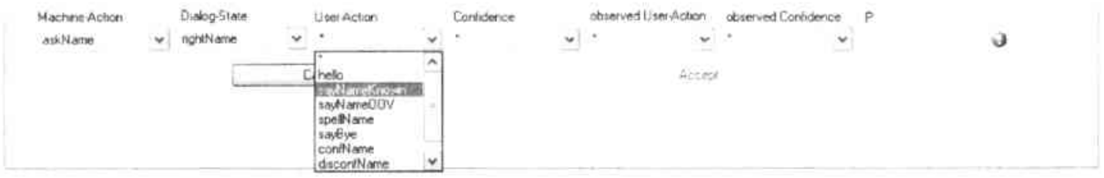


Figure 4.4: POMDP-Builder: Observation model



Machine Action	Dialog-State	User Action	Confidence	observed User-Action	observed Confidence	P
askName	rightName	*	*	*	*	

Figure 4.5: POMDP-Builder: Observation model (add)

The reward model is shown in a table on the fourth tab-page (fig. 4.6). Again, the items can be edited, deleted and rearranged. The add-process in the reward model is also like the add-process in the transition model and the observation model. It is shown in figure 4.7.

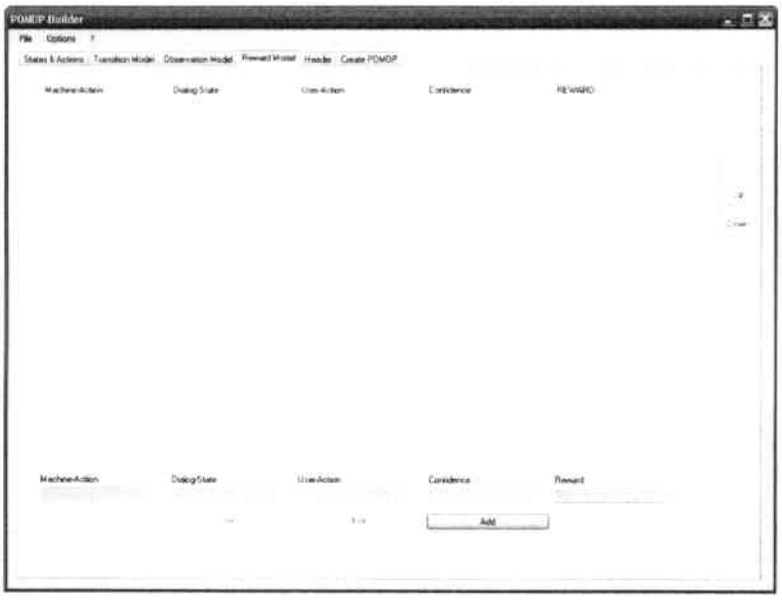


Figure 4.6: POMDP-Builder: Reward model

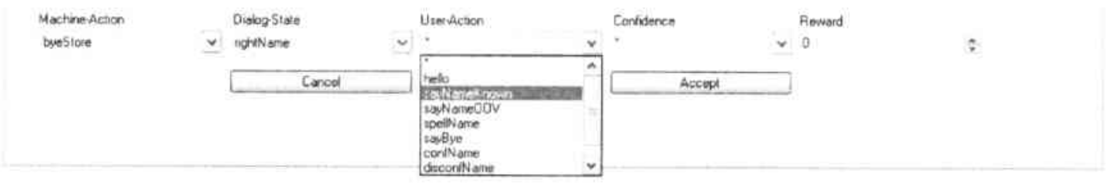


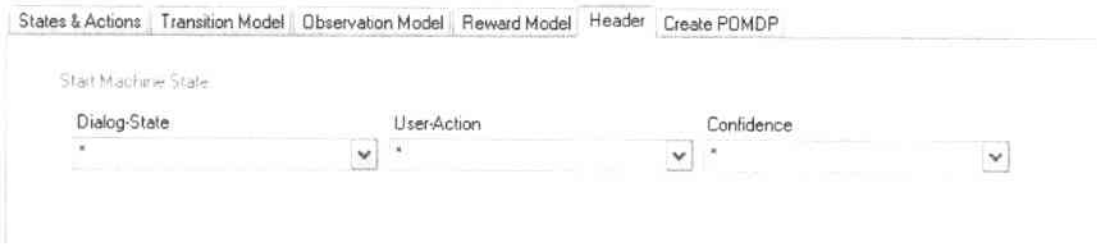
Figure 4.7: POMDP-Builder: Reward model (add)

The initial belief-state can be selected on the fifth tab-page (fig. 4.8).

On the sixth tab-page, the designer can select an output file. Then POMDP-Builder creates a pomdp-file in Tony Cassandra's format according to the descriptions the designer has made. The interface is shown in figure 4.9.

Wildcard

The description of a POMDP using the POMDP-Builder is still very long. Therefore, an addition to the standard POMDP-Builder is the use of the wildcard '*'.



Dialog-State	User-Action	Confidence
*	*	*

Figure 4.8: POMDP-Builder: Initial belief-state

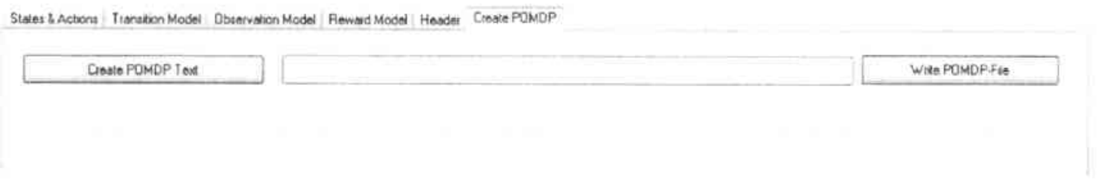
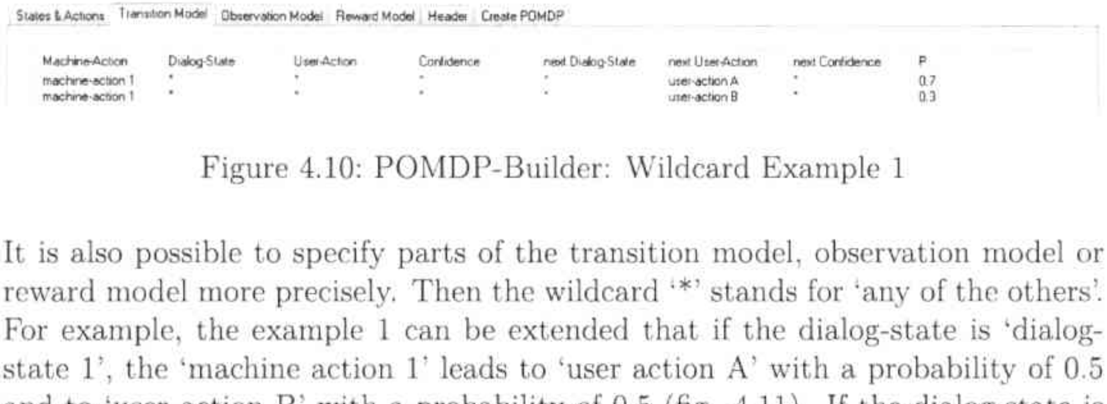


Figure 4.9: POMDP-Builder: Create pomdp-file

The designer can select this wildcard in each drop-down-list. One interpretation of the wildcard '*' is 'any'. This wildcard allows describing parts of the transition model, observation model or reward model independent of the other parts. For example, it allows a connection between the machine action and the user action in the transition model. Figure 4.10 shows an example where the 'machine action 1' leads to 'user action A' with a probability of 0.7 and to 'user action B' with a probability of 0.3.



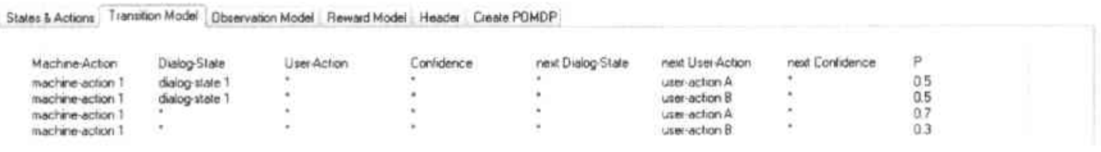
Machine-Action	Dialog-State	User-Action	Confidence	next Dialog-State	next User-Action	next Confidence	P
machine-action 1	*	*	*	*	user-action A	*	0.7
machine-action 1	*	*	*	*	user-action B	*	0.3

Figure 4.10: POMDP-Builder: Wildcard Example 1

It is also possible to specify parts of the transition model, observation model or reward model more precisely. Then the wildcard '*' stands for 'any of the others'. For example, the example 1 can be extended that if the dialog-state is 'dialog-state 1', the 'machine action 1' leads to 'user action A' with a probability of 0.5 and to 'user action B' with a probability of 0.5 (fig. 4.11). If the dialog-state is not 'dialog-state 1', the POMDP works exactly like the POMDP in example 1.

4.3 The Scenario

The idea is that the dialog takes place in a multimodal environment where a humanoid robot is talking to the user like the humanoid robot used in [Holzapfel and Waibel, 20



Machine-Action	Dialog-State	User-Action	Confidence	next Dialog-State	next User-Action	next Confidence	P
machine-action 1	dialog-state 1	*	*	*	user-action A	*	0.5
machine-action 1	dialog-state 1	*	*	*	user-action B	*	0.5
machine-action 1	*	*	*	*	user-action A	*	0.7
machine-action 1	*	*	*	*	user-action B	*	0.3

Figure 4.11: POMDP-Builder: Wildcard Example 2

This humanoid robot has standard perceptual components. Video cameras collect video data and microphones collect audio data. The video data can be used for face detection and face identification (FaceID). The audio data can be used for voice identification (VoiceID) and speech recognition. FaceID, VoiceID and dialog information can be integrated to the multimodal user ID [Holzapfel and Waibel, 2008b]. Automatic speech recognition extracts the semantic information from the audio data. The semantic information is mapped on a user action. This user action combined with the confidence of the multimodal user ID is the observation. The POMDP can now calculate its new belief-state and can select the best machine action according to the policy. This machine action is transformed into text and this text is transformed to speech for the user. This progress is shown in fig. 4.12.

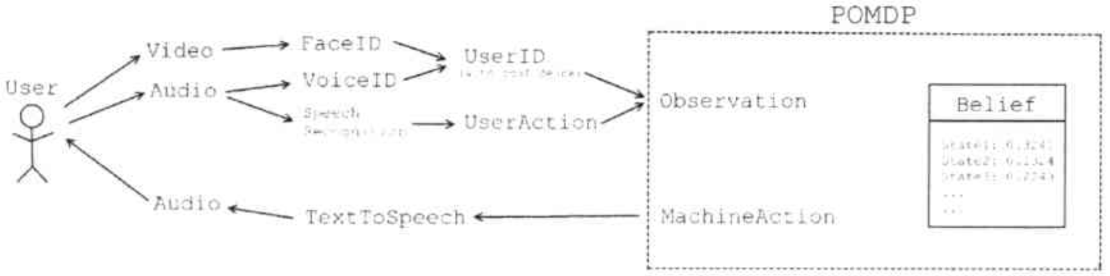


Figure 4.12: The Scenario Process

4.4 Person Identification Dialog

A person identification dialog is a dialog with the goal to identify the user. A person is identified by a name in a dialog. Therefore, the main goal is to get the correct name from the user. Thus, the machine has to be able to ask for the name of the user. Then the user would normally reply by saying his/her name. It is possible that the name the user said is not in the vocabulary (out of vocabulary, OOV). The machine has to be able to ask for the spelling of the name in order to learn even unknown names. When the machine has a name of the user but it is not sure if the name is correct, it should be able to ask for a confirmation. The user will then confirm or disconfirm the name. The machine can decide at the end of the dialog to store the result or to drop it. The user should be able to end the dialog at any time. The machine gets a confidence value, like the confidence from the multimodal user ID [Holzapfel and Waibel, 2008b], at each turn of the dialog. This confidence indicates the likelihood that the name the machine has is correct.

5 Name-Dialogs - Building, Solving and Results

This chapter describes the implementation of two person identification dialogs. The first dialog uses only one name (*Single-Name-Dialog*). This dialog is also the foundation for the second dialog which deals with the first and last name of the user (*Full-Name-Dialog*). The POMDP-Builder is used for the description of the POMDPs. ZMDP [Smith, 2006] is used for the solving because [Bui et al., 2007b] has shown that ZMDP handles complex POMDPs better than Perseus. Then the results are tested with the interactive simulator from the POMDP Toolkit [Bui et al., 2007a].

5.1 Creating the POMDP Description

The first part of a POMDP description is the definition of the machine actions, the user actions, the confidence values and the dialog-states. The second part is the definition of the transition model, observation model and reward model.

5.1.1 Single-Name-Dialog

The aim of the Single-Name-Dialog is to create a simple person identification dialog. This dialog deals only with one name for each user.

Machine-Actions

The machine can ask for the name of the user (*askName*). It can ask for a confirmation of the name (*askConfName*). Furthermore, it can ask for the spelling of the user's name (*askSpelling*). The machine can also end the dialog in two ways. It can end the dialog and store the result (*endStore*) or it can abort the dialog (*endAbort*). The machine can also execute an action which starts the dialog (*machineStart*). These machine actions are listed in table 5.1.

User-Actions

The user actions are listed in table 5.2.

machine action	description
machineStart	The machine starts the dialog.
askName	The machine asks the name of the user.
askConfName	The machine asks for the confirmation of the name.
askSpelling	The machine asks for the spelling of the user's name.
endStore	The machine ends the dialog successful.
endAbort	The machine aborts the dialog.

Table 5.1: Single-Name-Dialog - Machine-Actions

user action	description
userStart	The user starts the dialog.
sayNameKnown	The user says a known name.
sayNameOOV	The user says a name which is out of vocabulary.
spellName	The user spells his/her name.
confName	The user confirms the name.
disconfName	The user disconfirms the name.
userEnd	The user ends the dialog.

Table 5.2: Single-Name-Dialog - User-Actions

Confidence Values

The confidence needs to be discretized. The more confidence values are used, the more nuances can be differentiated. But more confidence values create a more complex POMDP. Some tests have indicated that six confidence values (table 5.3) are a good compromise. The lowest and the highest confidence cover 20 percent and each of the four values in the middle cover 15 percent.

confidence values	description
$c10$	The confidence is between 0 and 20 percent.
$c30$	The confidence is between 20 and 35 percent.
$c40$	The confidence is between 35 and 50 percent.
$c60$	The confidence is between 50 and 65 percent.
$c70$	The confidence is between 65 and 80 percent.
$c90$	The confidence is between 80 and 100 percent.

Table 5.3: Single-Name-Dialog - Confidence Values

Dialog-States

The dialog-state in the Single-Name-Dialog only deals with the name the machine got from the user. Four different states are distinguished: noName, rightName, wrongName and oovName. These dialog-states are explained in table 5.4.

dialog-state	description
$noName$	The machine has no name of the user.
$rightName$	The machine has the name of the user.
$wrongName$	The machine has a wrong name.
$oovName$	The machine has a user's name which is out of vocabulary.

Table 5.4: Dialog-States - Single-Name-Dialog

Transition Model

The description of the transition model is a difficult part. The use of the wildcard '*' makes the description easier and shorter. But many attempts have shown that it is still very complex and many errors are possible. One main difficulty is that not only the expectation of the user's behavior is modeled by the transition model but also the user himself in the training of the POMDP. The more complex the transition model becomes, the more sources of errors exist. Several tests have shown unintentional results. The fine tuning of a complex transition model is also very difficult.

To visualize the transition model of the single-name-dialog, it is divided into two parts. Part one (fig. 5.1) describes which user action can lead to which dialog state. Part two (fig. 5.2) describes which machine action can lead to which user action. Notice that the transition model probabilities depend on the confidence. Furthermore, some very unlikely transitions are left out to improve the clarity.

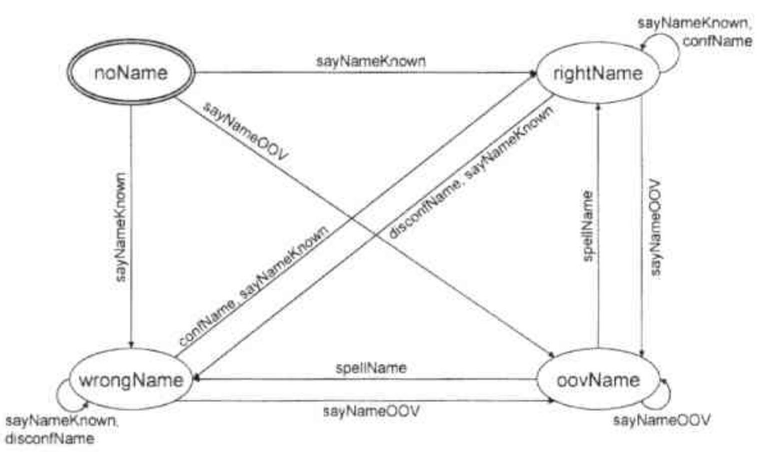


Figure 5.1: Simplified Transition Model - Part 1

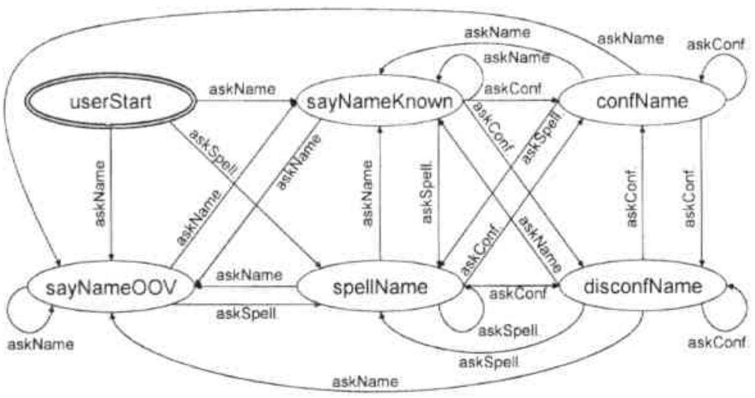


Figure 5.2: Simplified Transition Model - Part 2

Observation Model

A simple observation model was chosen for this dialog. The factored architecture allows a direct connection between the state and the observation. The main question is whether the observations are correct.

The observed confidence directly implies the current confidence because the confidence is directly transmitted to the machine. The user action $sayNameKnown$ indicates that the internal name slot has been filled and there is also no uncertainty. When the user says a name which is out of vocabulary, the name slot has not been filled. If the user spells his/her name, the name slot will be filled by the name spelling unit. The start and the end of the person identification dialog are also certain. Only the user actions $confName$ and $disconfName$ are uncertain. Since the other user actions are certain, $confName$ can only be $confName$ or $disconfName$ and $disconfName$ can only be $disconfName$ or $confName$. It is most likely that the observation is correct ($P = 0.9$). The probability that the observation was not correct is 0.1.

Reward Model

The reward model has a strong influence on the solving process and therefore on the result. Most parts of a reward model depend on psychological values. Obviously, the main target is to create a dialog which is convenient for user and leads to the right user's name.

The following reward model has been chosen. The main goal of the dialog is to get the right name from the user and then store the result. Therefore, the machine action *endStore* with the dialog-state $rightName$ gives a reward of 20. The machine action *endStore* in any other dialog-state has a reward of -20. The use of the machine action *endAbort* has always a reward of -10. The idea is that the machine avoids using *endAbort*, but it will use *endAbort* to end the dialog when the dialog-state is not rightName. Since the question for confirmation of a name cannot be formulated if the dialog-state is $noName$ or $oovName$, it has a reward of -10. The question for the spelling is unusual and inconvenient in a normal dialog and the machine should avoid this question unless it is necessary. Therefore, the machine action *askSpelling* has a reward of -5. Each other machine action and state combination has a reward of -1 because the machine should try to keep the dialog short.

Initial Belief-State

The initial belief-state is the dialog-state $noName$ and the user action $userStart$ uniformly distributed over all confidence values.

5.1.2 Full-Name-Dialog

The aim of the Full-Name-Dialog is to create a person identification dialog which is more realistic than the Single-Name-Dialog. This dialog deals with one first name and one last name for each user.

Machine-Actions

The machine actions are more specific in the Full-Name-Dialog. People are identified by their first and last name. Therefore, the machine has to be able to ask for the first name and the last name separately. This leads to more machine actions and therefore a much more complex system. The machine actions of the Full-Name-Dialog are listed in table 5.5.

machine action	description
$machineStart$	The machine starts the dialog.
$askFirst$	The machine asks the first name of the user.
$askLast$	The machine asks the last name of the user.
$askConfFirst$	The machine asks for the confirmation of the first name.
$askConfLast$	The machine asks for the confirmation of the last name.
$askSpellingFirst$	The machine asks for the spelling of the user's first name.
$askSpellingLast$	The machine asks for the spelling of the user's last name.
$endStore$	The machine ends the dialog successful.
$endAbort$	The machine aborts the dialog.

Table 5.5: Full-Name-Dialog - Machine-Actions

User-Actions

The user actions are listed in table 5.6.

Confidence Values

The confidence values used in the Full-Name-Dialog are the same values that were used in the Single-Name-Dialog.

Dialog-States

Since the Full-Name-Dialog deals with two names, information about the state of these two names is needed. The states of the first name and the last name

user action	description
userStart	The user starts the dialog.
sayFirstKnown	The user says a known first name.
sayLastKnown	The user says a known last name.
sayFirstOOV	The user says a first name which is out of vocabulary.
sayLastOOV	The user says a last name which is out of vocabulary.
spellFirst	The user spells his/her first name.
spellLast	The user spells his/her last name.
confFirst	The user confirms the first name.
confLast	The user confirms the last name.
disconfFirst	The user disconfirms the first name.
disconfLast	The user disconfirms the last name.
userEnd	The user ends the dialog.

Table 5.6: Full-Name-Dialog - User-Actions

confidence values	description
$c10$	The confidence is between 0 and 20 percent.
$c30$	The confidence is between 20 and 35 percent.
$c40$	The confidence is between 35 and 50 percent.
$c60$	The confidence is between 50 and 65 percent.
$c70$	The confidence is between 65 and 80 percent.
$c90$	The confidence is between 80 and 100 percent.

Table 5.7: Full-Name-Dialog - Confidence Values

are each like the state in the Single-Name-Dialog. Thus, the dialog-states in the Full-Name-Dialog are the Cartesian product of the dialog-states of the first name and the dialog-states of the last name. These dialog-states are listed in table 5.8 and table 5.9.

dialog-state	description
$noFirst$	The machine has no first name of the user.
$rightFirst$	The machine has the first name of the user.
$wrongFirst$	The machine has a wrong first name.
$oovFirst$	The machine has a user's first name which is out of vocabulary.

Table 5.8: Full-Name-Dialog - Dialog-States (First Name)

dialog-state	description
$noLast$	The machine has no last name of the user.
$rightLast$	The machine has the last name of the user.
$wrongLast$	The machine has a wrong last name.
$oovLast$	The machine has a user's last name which is out of vocabulary.

Table 5.9: Full-Name-Dialog - Dialog-States (Last Name)

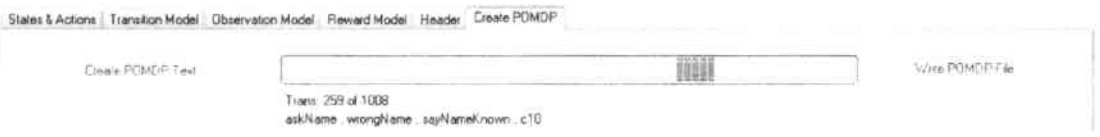
Transition Model, Observation Model, Reward Model

The transition model, observation model and reward model of the Full-Name-Dialog are based on the models of the Single-Name-Dialog. It is like two Single-Name-Dialogs in principle. But the description is much more complex because it contains much more states and actions.

5.2 Solving the POMDP

5.2.1 Single-Name-Dialog

The pomdp-file for the Single-Name-Dialog has been created (fig. 5.3) successfully after the description. Then, the ZMDP solver can read the pomdp-file to solve the Single-Name-Dialog. The solver creates a policy file which can be used to calculate the best machine action for each situation. The ZMDP solver needs only about a minute to solve the Single-Name-Dialog.



Trans: 259 of 1008
askName . wrongName . sayNameKnown . c10

Figure 5.3: POMDP-Builder: Single-Name-Dialog - Creating POMDP-File

5.2.2 Full-Name-Dialog

The increased complexity of the POMDP is also shown by the creation of the pomdp-file. It takes more time to build the pomdp-file of the Full-Name-Dialog than it took in the Single-Name-Dialog. This is also shown by the size of the created file. It has a size of more than 200 MBytes while the pomdp-file of the Single-Name-Dialog has only a size of about 2 MBytes.

A question was: Can the ZMDP solver handle the Full-Name-Dialog? ZMDP can read in the pomdp-file and it can start solving it. But it takes a long time and the longer ZMDP runs the more main memory it consumes. In this test, the ZMDP solver was terminated after three days due to memory restrictions. The ZMDP solver could not create a fully trained policy file. The partially trained policy file is discussed in 5.3.4. It might be possible to solve the Full-Name-Dialog in a few years on a computer with more main memory. Until the solving of the Full-Name-Dialog is possible, two Single-Name-Dialogs could be used to get the first and last name from the user.

5.3 Evaluation and Results

5.3.1 Metric

Number of Entries

The function $E(M, D)$ describes how many entries the transition model, observation model or reward model has in a specified POMDP description. The parameter M indicates the model (transition, observation, reward) and the parameter D stand for the POMDP description. The function $E(M, D)$ is used to compare the extent of different descriptions for the same POMDP.

5.3.2 Evaluation: POMDP Description

In this section, the function $E(\cdot, \cdot)$ (see 5.3.1) is used to compare POMDP descriptions of the POMDP Builder with POMDP descriptions in the pomdp-file format.

Single-Name-Dialog

The transition model of the POMDP-Builder has 403 entries to describe the Single-Name-Dialog. The pomdp-file has 18,648 entries in the transition model. The difference is more obvious in the observation model and the reward model. The POMDP-Builder has 15 entries in the observation model and the pomdp-file has 1,296 entries. In the reward model, the POMDP-Builder has 11 entries and the pomdp-file has 1,008 entries. These results are shown in table 5.10. The POMDP description of the POMDP Builder is significantly smaller than the POMDP description in the pomdp-file format.

$E(\cdot,\cdot)$	POMDP-Builder	pomdp-file	*factor*
Transition Model	403	18,648	46.3
Observation Model	15	1,296	86.4
Reward Model	11	1,008	91.6

Table 5.10: Single-Name-Dialog - Comparison (Number of entries)

Full-Name-Dialog

The transition model of the POMDP-Builder has 5,409 entries while the pomdp-file has 2,224,800 entries in the transition model. Also, the observation model and the reward model are much smaller using the POMDP-Builder. The observation model needed 18 entries in the POMDP-Builder and 10,368 entries in the pomdp-file. The reward model needed 22 entries in the POMDP-Builder and 10,368 entries in the pomdp-file. These results are also shown in table 5.11. The advantage of the POMDP-Builder is much more significant in this complex example than in the Single-Name-Dialog.

$E(\cdot,\cdot)$	POMDP-Builder	pomdp-file	*factor*
Transition Model	5,409	2,224,800	411.3
Observation Model	18	10,368	576.0
Reward Model	22	10,368	471.3

Table 5.11: Full-Name-Dialog - Comparison (Number of entries)

5.3.3 Evaluation: Dialog

Single-Name-Dialog

Six typical test cases are selected. They have been chosen because they have been observed in dialogs of this kind. These test cases have been used to compare the Single-Name-Dialog with a dialog with no confidence information. The selected test cases differ in whether the user is known and whether the first hypothesis is right, wrong or indicate an OOV name. The confidence for the Single-Name-Dialog depends on whether the user is known or unknown and the first hypothesis is right or wrong. In cases 5 and 6, the correct name of the user is out of vocabulary (OOV). In all cases, it is assumed that the second hypothesis the machine gets is always right. The test cases are listed in table 5.12.

case #	user	first hypothesis	confidence	correct name OOV?
1	known	right	high	no
2	known	wrong	low	no
3	unknown	right	low	no
4	unknown	wrong	low	no
5	unknown	wrong	low	yes
6	unknown	OOV	low	yes

Table 5.12: Single-Name-Dialog - Test Cases

Test Case 1 In test case 1, the user is known to the machine and the first hypothesis is right. Under these conditions, it is most likely that the confidence is high. Test case 1 shows that the machine in the Single-Name-Dialog does not ask for a confirmation when the confidence is very high. Therefore, the Single-Name-Dialog (tab. 5.13) needs only one turn whereas the dialog with no confidence (tab. 5.14) needs two turns in test case 1.

Machine:	askName
User:	sayNameKnown-c90
Machine:	endStore

Table 5.13: Case 1 - Single-Name-Dialog

Machine:	askName
User:	sayNameKnown
Machine:	askConfName
User:	confName
Machine:	endStore

Table 5.14: Case 1 - no confidence

Test Case 2 In test case 2, the user is known to the machine and the first hypothesis is wrong. It is most likely that the confidence is low under these conditions. Test case 2 shows the machine in the Single-Name-Dialog (tab. 5.15) does not ask for a confirmation when it gets a name with a very low confidence. Instead, it asks for the spelling of the user's name to get a new hypothesis. The machine in the dialog without confidence (tab. 5.16) asks two times for the name and the confirmation. Therefore, the Single-Name-Dialog takes three turns while the dialog without confidence needs four.

Machine:	askName
User:	sayNameKnown-c10
Machine:	askSpelling
User:	spellName-c10
Machine:	askConfName
User:	confName-c10
Machine:	endStore

Table 5.15: Case 2 - Single-Name-Dialog

Machine:	askName
User:	sayNameKnown
Machine:	askConfName
User:	disconfName
Machine:	askName
User:	sayNameKnown
Machine:	askConfName
User:	confName
Machine:	endStore

Table 5.16: Case 2 - no confidence

Test Case 3 In test case 3, the user is unknown to the machine and the first hypothesis is right. When the user is unknown, it is most likely that the confidence is low. Test case 3 shows, the machine in the Single-Name-Dialog (tab. 5.17) does not ask for a confirmation because the confidence is very low. Instead, it asks for the spelling of the user's name. The machine in the dialog without confidence (tab. 5.16) asks for the confirmation. Therefore, the Single-Name-Dialog takes three turns while the dialog without confidence needs two.

Machine:	askName
User:	sayNameKnown-c10
Machine:	askSpelling
User:	spellName-c10
Machine:	askConfName
User:	confName-c10
Machine:	endStore

Table 5.17: Case 3 - Single-Name-Dialog

Machine:	askName
User:	sayNameKnown
Machine:	askConfName
User:	confName
Machine:	endStore

Table 5.18: Case 3 - no confidence

Test Case 4 In test case 4, the user is unknown to the machine and the first hypothesis is wrong. It is most likely that the confidence is low because the user is unknown. In test case 4, the machine in the Single-Name-Dialog (tab. 5.19) does not ask for a confirmation because the confidence is very low, but it asks for the spelling of the user's name. The machine in the dialog without confidence (tab. 5.20) asks two times for the name and the confirmation. Therefore, the Single-Name-Dialog takes three turns while the dialog without confidence needs four.

Machine:	askName
User:	sayNameKnown-c10
Machine:	askSpelling
User:	spellName-c10
Machine:	askConfName
User:	confName-c10
Machine:	endStore

Table 5.19: Case 4 - Single-Name-Dialog

Machine:	askName
User:	sayNameKnown
Machine:	askConfName
User:	disconfName
Machine:	askName
User:	sayNameKnown
Machine:	askConfName
User:	confName
Machine:	endStore

Table 5.20: Case 4 - no confidence

Test Case 5 In test case 5, the user is unknown to the machine, the first hypothesis is wrong and the correct user name is out of vocabulary. It is most likely that the confidence is low because the user is unknown. In test case 5, the machine in the Single-Name-Dialog (tab. 5.21) again does not ask for a confirmation because the confidence is very low. Instead, it asks for the spelling of the user's name. The machine in the dialog without confidence (tab. 5.22) asks two times for the name and the confirmation and one time for the spelling. Therefore, the Single-Name-Dialog takes three turns while the dialog without confidence needs five.

Test Case 6 In test case 6, the user is unknown to the machine, the first hypothesis is out of vocabulary and the correct user is also out of vocabulary. It is most likely that the confidence is low because the hypothesis is OOV. Test case 6 shows that the Single-Name-Dialog (tab. 5.23) and the dialog without confidence (tab. 5.24) react in the same way. Both ask for the spelling and then for the confirmation. Thus, both dialogs need three turns.

Test Case Summary The test cases have shown that the Single-Name-Dialog needs less turns in most cases (tab. 5.25). The dialog without confidence is faster

Machine:	askName
User:	sayNameKnown-c10
Machine:	askSpelling
User:	spellName-c10
Machine:	askConfName
User:	confName-c10
Machine:	endStore

Table 5.21: Case 5 - Single-Name-Dialog

Machine:	askName
User:	sayNameKnown
Machine:	askConfName
User:	disconfName
Machine:	askName
User:	sayNameOOV
Machine:	askSpelling
User:	spellName
Machine:	askConfName
User:	confName
Machine:	endStore

Table 5.22: Case 5 - no confidence

Machine:	askName
User:	sayNameOOV-c10
Machine:	askSpelling
User:	spellName-c10
Machine:	askConfName
User:	confName-c10
Machine:	endStore

Table 5.23: Case 6 - Single-Name-Dialog

Machine:	askName
User:	sayNameOOV
Machine:	askSpelling
User:	spellName
Machine:	askConfName
User:	confName
Machine:	endStore

Table 5.24: Case 6 - no confidence

only in test case 3 (user: unknown, first hypothesis: right). These test cases of basic dialogs show the advantage of the presented system with confidence values.

case #	Turns (Single-Name-Dialog)	Turns (Dialog without Confidence)
1	1	2
2	3	4
3	3	2
4	3	4
5	3	5
6	3	3

Table 5.25: Test Cases - Summary

5.3.4 Discussion

Single-Name-Dialog

The policy from the Single-Name-Dialog has been tested using the interactive simulator of the POMDP Toolkit. The machine starts the dialog by asking the name of the user. Then the machine reacts very well when it observes a known name or an OOV name. This can be seen in the first two dialogs (table 5.26 and 5.27). Especially the handling of a high confidence is good since the machine does not ask for a confirmation (dialog 3, tab. 5.28). This example shows that the machine tries to keep the dialog short. But the dialog is not always short. Multiple tests have shown that the machine never aborts the dialog by itself. It would be desirable that the machine ends the dialog if it has no result after a certain amount of turns. But the POMDP model of the Single-Name-Dialog does not allow that. The dialog-state does not hold any information about the length of the current dialog. But with such a count the training of the POMDP would probably be much more difficult because this would lead to a different state in each situation depending on the current length of the dialog. But it still would be interesting to test a system with this possibility in future work. Dialog 4 (tab. 5.29) shows that the machine also asks for the spelling when the confidence is very low. This seems to be a good choice in that situation. The confidence is very low and it is most likely that the user will disconfirm the hypothesis of the name recognition unit. The spelling unit will give new information and can therefore lead to a better result in this situation. Dialog 5 (tab. 5.30) shows a situation where the machine asks for the name and the spelling in turns. The name recognition unit gets an OOV name, the spelling was disconfirmed and then the name recognition unit tries again to get the right name. If the name which the user is trying to say is really out of vocabulary, this strategy is not good because the name recognition unit can never get the correct name. But this strategy makes sense since the name can wrongly be recognized as out of vocabulary. Overall, the Single-Name-Dialog works and can be used as basis for the Full-Name-Dialog.

Machine:	askName
User:	sayNameKnown-c60
Machine:	askConfName
User:	confName-c60
Machine:	endStore

Table 5.26: Single-Name-Dialog - Dialog 1

Machine:	askName
User:	sayNameOOV-c60
Machine:	askSpelling
User:	spellName-c60
Machine:	askConfName
User:	confName-c60
Machine:	endStore

Table 5.27: Single-Name-Dialog - Dialog 2

Machine:	askName
User:	sayNameKnown-c90
Machine:	endStore

Table 5.28: Single-Name-Dialog - Dialog 3

Machine:	askName
User:	sayNameKnown-c10
Machine:	askSpelling
User:	spellName-c10
Machine:	askConfName
User:	confName-c10
Machine:	endStore

Table 5.29: Single-Name-Dialog - Dialog 4

Machine:	askName
User:	sayNameOOV-c30
Machine:	askSpelling
User:	spellName-c30
Machine:	askConfName
User:	disconfName-c30
Machine:	askName
User:	sayNameOOV-c30
Machine:	askSpelling
User:	spellName-c30
Machine:	askConfName
User:	confName-c30
Machine:	endStore

Table 5.30: Single-Name-Dialog - Dialog 5

Full-Name-Dialog

The first tests have shown that the policy file of the Full-Name-Dialog only works partly. The machine starts by asking the first name. If the user says a known first name with a confidence $c40$ or higher, the dialog part for the first name behaves exactly like the Single-Name-Dialog. The machine tries to achieve the last name after the first name is confirmed. If the last observed confidence has been $c70$ or higher the machine asks for the last name (dialog 1, tab. 5.31). When the machine asks for the last name, it handles OOV names without any problem (dialog 2, tab. 5.32). If the last observed confidence was $c60$ or lower the machine asks for the spelling of the last name (dialog 3, tab. 5.33). This behavior is not wanted since directly asking for the spelling is not natural and it is very likely to get the wrong spelling. But the behavior of the machine is worse when the user says a known name with a confidence of $c30$ or $c10$ and disconfirms the first hypothesis. Then the machine asks for the spelling of the *last* name and tries to confirm the last name. When the machine finally has a confirmed last name, it asks for the confirmation of the first name, which was already disconfirmed. The machine keeps asking for a confirmation of the first name even if the user disconfirms the first name again (dialog 4, tab. 5.34). The problem is that the machine selects the machine action $askConfFirst$ when the belief-state indicates that the current dialog-state is $wrongFirst\text{-}rightLast$, the current user action is $confLast$ or $disconfFirst$ and the current confidence is $c10$ or $c30$. The machine asks for the first name again, only when the confidence changes to $c40$ or higher. This is clearly unacceptable. But the problem might not exist when the solver can run longer. The worst behavior shows the machine when it observes the user action $sayFirstOOV$ at the beginning. Then the machine chooses the machine action *machineStart*. When the user replies by *userStart* with a confidence of $c60$ or higher, the machine asks again for the first name (dialog 5, tab. 5.35). Also, this behavior might be explained by the insufficient training. Dialog 6 (tab. 5.36) shows that the machine can handle an OOV first name when it already has a confirmed last name. All in all the results of the Full-Name-Dialog show that this policy cannot be used in a real dialog.

Machine:	askFirst
User:	sayFirstKnown-c60
Machine:	askConfFirst
User:	confFirst-c70
Machine:	askLast
User:	sayLastKnown-c90
Machine:	endStore

Table 5.31: Full-Name-Dialog - Dialog 1

Machine:	askFirst
User:	sayFirstKnown-c70
Machine:	askConfFirst
User:	confFirst-c70
Machine:	askLast
User:	sayLastOOV-c70
Machine:	askSpellingLast
User:	spellLast-c70
Machine:	askConfLast
User:	disconfLast-c70
Machine:	askLast
User:	sayLastOOV-c70
Machine:	askSpellingLast
User:	spellLast-c70
Machine:	askConfLast
User:	confLast-c70
Machine:	endStore

Table 5.32: Full-Name-Dialog - Dialog 2

Machine:	askFirst
User:	sayFirstKnown-c30
Machine:	askConfFirst
User:	confFirst-c30
Machine:	askSpellingLast
User:	spellLast-c30
Machine:	askConfLast
User:	disconfLast-c30
Machine:	askSpellingLast
User:	spellLast-c30
Machine:	askConfLast
User:	confLast-c30
Machine:	endStore

Table 5.33: Full-Name-Dialog - Dialog 3

Machine:	askFirst
User:	sayFirstKnown-c30
Machine:	askConfFirst
User:	disconfFirst-c30
Machine:	askSpellingLast
User:	spellLast-c30
Machine:	askConfLast
User:	confLast-c30
Machine:	askConfFirst
User:	disconfFirst-c30
Machine:	askConfFirst
User:	disconfFirst-c30
Machine:	askConfFirst
...	...

Table 5.34: Full-Name-Dialog - Dialog 4

Machine:	askFirst
User:	sayFirstOOV-c60
Machine:	machineStart
User:	userStart-c60
Machine:	askFirst
User:	sayFirstOOV-c60
Machine:	machineStart
User:	userStart-c60
Machine:	askFirst
...	...

Table 5.35: Full-Name-Dialog - Dialog 5

Machine:	askFirst
User:	sayFirstOOV-c30
Machine:	machineStart
User:	userStart-c30
Machine:	askSpellingLast
User:	spellLast-c30
Machine:	askConfLast
User:	confLast-c30
Machine:	askFirst
User:	sayFirstOOV-c30
Machine:	askSpellingFirst
User:	spellFirst-c30
Machine:	askConfFirst
User:	confFirst-c30
Machine:	endStore

Table 5.36: Full-Name-Dialog - Dialog 6

6 Conclusion

In this work, a person identification dialog with explicit confidence values has been built using a Partially Observable Markov Decision Process. For the description of the POMDP, it was necessary to develop the POMDP-Builder because the current description tools were insufficient. Two person identification dialogs with explicit confidence values (Single-Name-Dialog and Full-Name-Dialog) have been built using the POMDP-Builder. This has demonstrated that the description of a POMDP with explicit confidence values using the POMDP-Builder is possible. It has also been shown that the description of the POMDP using the POMDP-Builder is significantly smaller and therefore less complex than other more common description methods. The Single-Name-Dialog, which deals with one name of the user, was successfully solved by the ZMDP solver. The result was tested in several test cases. These test cases pointed out the advantage of the use of explicit confidence values. Especially in situations where the user is known to the system (i.e. name exists in the vocabulary and models are trained), the dialog requires less turns when using explicit confidence values. This shows that explicit confidence values are especially useful in person identification dialogs which often deal with the same persons. The Single-Name-Dialog delivers a very good result and shows the advantage of POMDPs: Realistic description and very good results. A larger dialog manager (Full-Name-Dialog) was also created using the POMDP-Builder. The Full-Name-Dialog deals with the first and the last name of the user. Although, the Full-Name-Dialog uses only one name more than the Single-Name-Dialog for each user, the description is significantly larger. As a result, the Full-Name-Dialog could not be solved completely by the ZMDP solver due to memory restrictions. This has shown the main problem of POMDPs: The complexity raises very fast and the POMDP may not be solvable.

Bibliography

[Bui et al., 2007a] Bui, T., van Schooten, B., and Hofs, D. (2007a). POMDP Toolkit. *http://wwwhome.cs.utwente.nl/ hofs/pomdp/.*

[Bui et al., 2007b] Bui, T., van Schooten, B., and Hofs, D. (2007b). Practical dialogue manager development using POMDPs. In Keizer, S., Bunt, H., and Paek, T., editors, *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, pages 215–218, PA, USA. The Association for Computational Linguistics.

[Cassandra, 1999] Cassandra, A. R. (1999). Tony Cassandra's POMDP file format. *http://www.cs.brown.edu/research/ai/pomdp/examples/pomdp-file-spec.html.*

[Holzapfel and Waibel, 2008a] Holzapfel, H. and Waibel, A. (2008a). Learning and verification of names with multimodal user ID in dialog. In *International Conference on Cognitive Systems (CogSys 2008)*, Karlsruhe, Germany.

[Holzapfel and Waibel, 2008b] Holzapfel, H. and Waibel, A. (2008b). Modelling multimodal user ID in dialogue. In *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE*, pages 113–116, Goa.

[Kaelbling et al., 1995] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1995). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

[Puterman, 1994] Puterman, M. L. (1994). *Markov decision processes.* Wiley.

[Smith, 2006] Smith, T. (2006). ZMDP software for POMDP and MDP planning. *http://www.cs.cmu.edu/ trey/zmdp/.*

[Smith, 2007] Smith, T. (2007). *Probabilistic Planning for Robotic Exploration.* PhD thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

[Spaan and Vlassis, 2005] Spaan, M. T. J. and Vlassis, N. (2005). Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195—-220.

[Williams et al., 2005] Williams, J. D., Poupart, P., and Young, S. (2005). Factored partially observable markov decision processes for dialogue management. In *Proc Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI)*, Edinburgh.

[Young, 2006] Young, S. (2006). Using POMDPs for dialog management. In *Spoken Language Technology Workshop, 2006. IEEE*, pages 8–13, Palm Beach.

List of Figures

List of Tables