Hierarchical Reinforcement Learning for Spoken Dialogue Systems

Heriberto Cuayáhuitl

Doctor of Philosophy

Institute for Communicating and Collaborative Systems
School of Informatics
University of Edinburgh

January 2009

Abstract

This thesis focuses on the problem of scalable optimization of dialogue behaviour in speech-based conversational systems using reinforcement learning. Most previous investigations in dialogue strategy learning have proposed flat reinforcement learning methods, which are more suitable for small-scale spoken dialogue systems.

This research formulates the problem in terms of Semi-Markov Decision Processes (SMDPs), and proposes two hierarchical reinforcement learning methods to optimize sub-dialogues rather than full dialogues. The first method uses a hierarchy of SMDPs, where every SMDP ignores irrelevant state variables and actions in order to optimize a sub-dialogue. The second method extends the first one by constraining every SMDP in the hierarchy with prior expert knowledge. The latter method proposes a learning algorithm called 'HAM+HSMQ-Learning', which combines two existing algorithms in the literature of hierarchical reinforcement learning. Whilst the first method generates fully-learnt behaviour, the second one generates semi-learnt behaviour. In addition, this research proposes a heuristic dialogue simulation environment for automatic dialogue strategy learning. Experiments were performed on simulated and real environments based on a travel planning spoken dialogue system. Experimental results provided evidence to support the following claims: First, both methods scale well at the cost of near-optimal solutions, resulting in slightly longer dialogues than the optimal solutions. Second, dialogue strategies learnt with coherent user behaviour and conservative recognition error rates can outperform a reasonable hand-coded strategy. Third, semi-learnt dialogue behaviours are a better alternative (because of their higher overall performance) than hand-coded or fully-learnt dialogue behaviours. Last, hierarchical reinforcement learning dialogue agents are feasible and promising for the (semi) automatic design of adaptive behaviours in larger-scale spoken dialogue systems.

This research makes the following contributions to spoken dialogue systems which learn their dialogue behaviour. First, the *Semi-Markov Decision Process (SMDP)* model was proposed to learn spoken dialogue strategies in a scalable way. Second, the concept of *partially specified dialogue strategies* was proposed for integrating simultaneously hand-coded and learnt spoken dialogue behaviours into a single learning framework. Third, an *evaluation with real users* of hierarchical reinforcement learning dialogue agents was essential to validate their effectiveness in a realistic environment.

Keywords: Speech-based human-machine communication, spoken dialogue systems, dialogue strategy design, dialogue simulation/optimization/evaluation/coherence, dialogue knowledge representation, stochastic sequential decision making, temporal abstraction, hierarchical control, prior expert knowledge, finite state machines, Markov decision processes (MDPs), Semi-Markov decision processes (SMDPs), flat reinforcement learning, and hierarchical reinforcement learning.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Heriberto Cuayáhuitl*)

Acknowledgements

I would like to thank my supervisor Steve Renals without whose experienced guidance this research work would not have been possible. I am also grateful for the feedback I received from my co-supervisors Oliver Lemon and Hiroshi Shimodaira.

I am very grateful for the funding I received from the following Mexican institutions: PROMEP[1] and the Autonomous University of Tlaxcala (UATx)[2]. I thank the members of staff from both institutions for their help with administrative issues – among many from whose help I benefitted are Serafin Ortíz, Roman Mendoza, Antonio Durante, Leticia Flores and Carlos Pérez.

I would like to thank those who shared their knowledge with me and enabled me to complete the work described in this thesis: Ben Serridge for introducing me to the fascinating world of speech technologies; the authors of papers and books cited in this thesis; to the University of Edinburgh and the School of Informatics for making available an excellent environment for research; to the staff of the Centre for Speech Technology Research (CSTR) and Human-Computer Research Centre (HCRC) for their help and friendship; to my fellow-students for our enriching discussions – Pei Yun Hsueh, Ivan Meza, Zhang Le, Markus Becker, and Verena Rieser; to the Dialogs on Dialogs reading group at CMU for their enriching meetings on Spoken Dialogue Systems; to Alan Smaill for giving me the opportunity to be tutor for his course on Fundamentals of Artificial Intelligence; to Mike Turnbull and Judy Cantley for their recommendations on writing style; to the testers of my spoken dialogue system for giving me many ideas to improve this kind of system; and to my thesis reviewers Johanna Moore and Steve Young whose comments enriched the correctness and clarity of this thesis.

I am also grateful to all the people who made my PhD studies a pleasant experience in Edinburgh and in Scotland. First of all, to Patricia Farfán and Abraham Cuayáhuitl for being very supportive in many different ways. I thank the Mexican community in Edinburgh including families Makita, Acosta and Bhattacharya for the great time we had together. I thank Goff and Judy Cantley for such great Ceilidh dances and walks around Edinburgh, and for hosting me while I was doing thesis corrections. I thank Arnold and Margaret Cody for their friendship and our lunches after mass in St Mary's Cathedral. I thank the folks of St Mary's Cathedral choir – I really enjoyed being part of this nice group of people.

[1] http://promep.sep.gob.mx
[2] www.uatx.mx

Last but not least, I thank my relatives in Mexico for being very supportive: my father Alfredo Cuayáhuitl, my sisters Minerva and Flor, my nieces Laura, Danely and Flor iveth, my nephew Victor Hugo, and my brothers-in-law. To all of you and to anybody I forgot to mention – thank you very much!

Heriberto Cuayáhuitl

To the memory of my mother and brother.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Spoken dialogue interaction has been suggested by researchers and practitioners as a promising alternative way of communication between humans and machines (Zue and Glass, 2000). A compelling motivation is the fact that conversational speech is the most natural, efficient, and flexible means of communication among human beings. Because of the complexity of human-human interaction, human-machine conversations need to be much simpler. In our contemporary world there are many machines used in our daily lives such as computers, telephones, cars, and robots. We may not want to talk to them all the time; however, the following are sample scenarios where a talking machine would be useful:

- while driving a car our eyes and hands are busy, but we may want to control the car's resources or access the internet;
- when many people call a company simultaneously to book a service or request information and have lengthy waits on the line due to busy human operators;
- when we have to do complex searches for information that depend on a dialogue history rather than on a single sentence and we only have a small keyboard;
- when giving instructions to a robot capable of a wide range of tasks;
- when a disabled person wants to interact with a machine;
- when a person does not want to use a keyboard.

Talking to a machine requires a spoken dialogue system. These systems may be alternatively referred to in the literature as 'conversational agents', 'spoken language systems' or 'conversational interfaces' (Jurafsky and Martin, 2008; Huang et al., 2001;

McTear, 2004). Such systems should be able to understand what a person says, take an appropriate action, and then provide a response. Ideally, spoken dialogue systems should yield successful, efficient and natural conversations within a given domain. However, building such systems is still a challenge for science and engineering.

A spoken dialogue system can be described as having four interlinked modules: speech recognition and understanding, a dialogue manager, language and speech generation, and knowledge base (Figure 1.1). It operates cyclically as follows: the user makes a verbal response and the corresponding speech wave is given to the speech recognition and understanding module to extract a compact representation (referred to as 'meaning') of what the user has said; such a meaning is used by the dialogue manager to choose an action based on the current dialogue history; the language and speech generation module takes that action so as to generate a spoken response. The cycle continues until one of the conversants (user or machine) terminates the dialogue. In addition, the knowledge base keeps track of all the information generated through the dialogue history, which is queried and/or updated by any of the system modules.

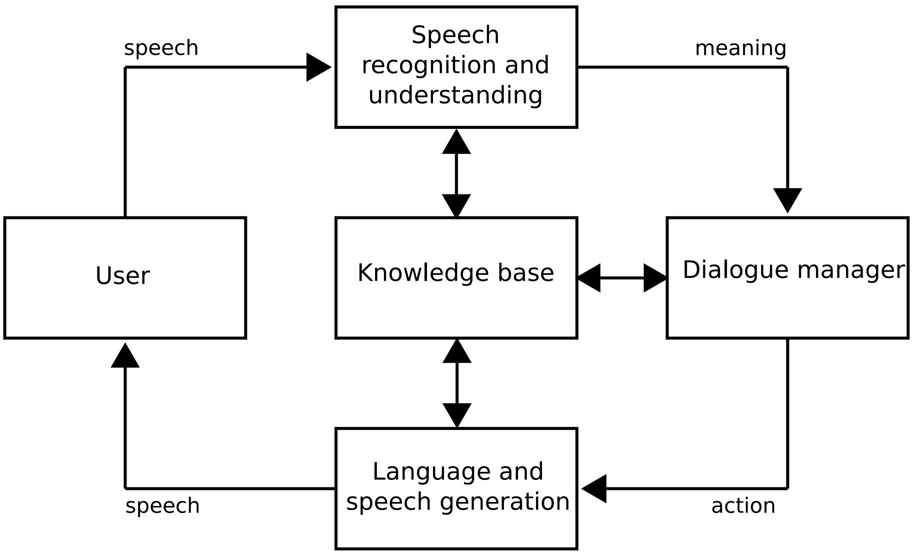


Figure 1.1: *A modular high-level architecture of a spoken dialogue system interacting with a user. This thesis focuses on the dialogue manager module.*

Although currently available human language technologies allow the building of working systems, they still face a number of problems and are likely to fail in the following situations: noisy environments, unknown vocabularies and meanings, unknown speech accents, requirements for world knowledge, or richer dialogue behaviour. This thesis is concerned with the design of spoken dialogue managers that are capable of learning to optimize their dialogue behaviour in a scalable and efficient way.

1.1 Motivation

Designing the behaviour of spoken dialogue managers for successful, efficient and natural conversations is a challenging goal. Dialogue managers behave by following a **dialogue strategy**, also referred to as 'dialogue policy' or 'dialogue behaviour'. Dialogue strategies are stochastic sequential decision makers as illustrated in Figure 1.2. For each situation (dialogue state) the strategy has to choose an action to change the current state – these transitions are stochastic because the dialogue state is not known with certainty. The task of the dialogue strategy is to choose appropriate actions for each possible dialogue state. Such strategies are typically hand-crafted by system designers. However, it turns out that this approach has a number of limitations: (1) it is not always easy to specify action-selection at some points in the dialogue (lack of optimization); (2) dialogue behaviour for the entire user population is generic and static (lack of adaptivity); (3) this is a labour-intensive task, especially for large systems.

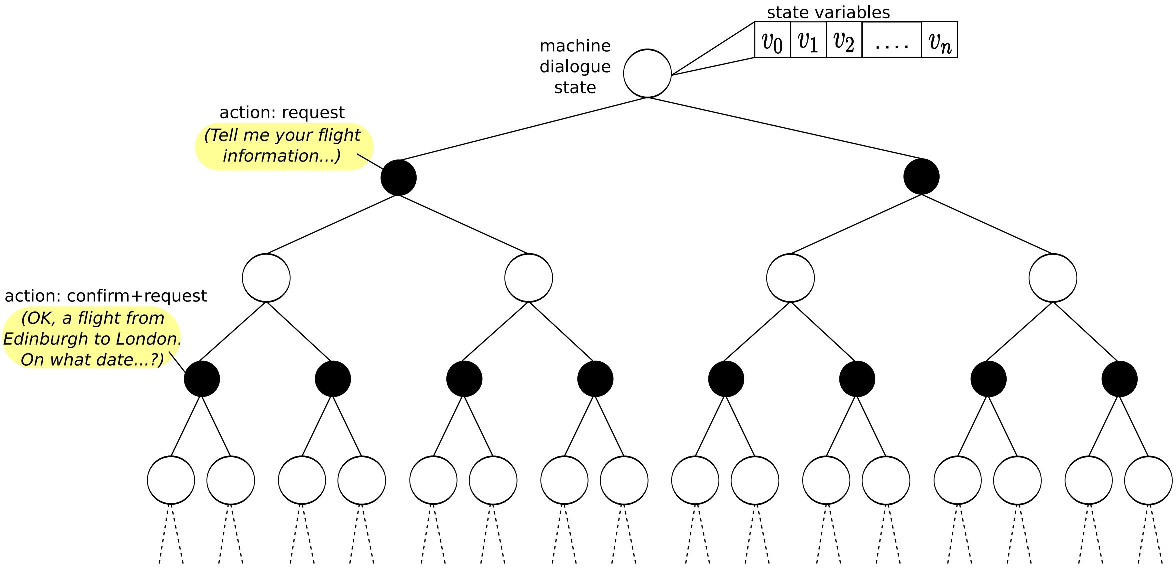


Figure 1.2: *Illustration of flat sequential decision-making for spoken dialogue. Empty circles are dialogue states and their possible transitions result from an executed action.*

As an alternative approach to hand-crafted design, Levin and Pieraccini (1997) framed the problem of dialogue strategy design as an optimization problem, and suggested MDP-based reinforcement learning for such a purpose. But it has proved difficult to develop spoken dialogue systems under this framework: two of the crucial issues are that of *uncertainty* and *scalability*. In the former, the dialogue states are assumed to be known with certainty; in the latter, the number of unique dialogue states grows exponentially as more information is incorporated. As an alternative approach, Roy et al. (2000) suggested the POMDP model to handle uncertainty in the conversation, but it has been difficult to apply this model to large-scale dialogue systems.

Previous work has optimized the behaviour of spoken dialogue systems for simple interactions using a single dialogue goal with only a few slots of information. The development and deployment of larger-scale systems remain as an important research avenue for their application in the real world. Proposing and evaluating a more scalable dialogue optimization framework is what has motivated this research.

Designing the behaviour of conversational agents in an automatic way matches the so called ‘rational agents’ also known as ‘intelligent agents’ and is central to artificial intelligence. Building and testing such kind of agents in large applications is of importance to the advancement of this research field, and are defined as follows.

> For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has (Russell and Norvig, 2003).

1.2 Research goal

This thesis investigates how to optimize the behaviour of spoken dialogue systems in a scalable, efficient and effective way under the reinforcement learning paradigm. It leaves aside the issue of uncertainty handling and focuses its attention on scaling the MDP-based reinforcement learning framework. For such a purpose this research aims to answer the following question: *How to learn dialogue strategies for large-scale information-seeking spoken dialogue systems?*

A solution to this problem would contribute towards the development of larger-scale spoken dialogue systems than those attempted so far. To that end three objectives are established. Firstly, to simulate and evaluate task-oriented and multi-goal human-machine conversations based on dialogue acts: this objective will be used to generate a large number of conversations for dialogue strategy learning in an automatic way. Secondly, to learn spoken dialogue strategies for large state-action spaces in order to scale up the MDP (Markov Decision Process) framework with a hierarchical approach (see next section). Although this research does not address the issue of uncertainty in the dialogue state, the resulting framework aims to provide relevant findings that may be used to scale up other models. Finally, to validate the findings from simulations by evaluating learnt dialogue strategies in a realistic environment: this objective used a real spoken dialogue system evaluated by real users.

1.3 Approach

Most previous work on dialogue strategy learning aimed for a single global solution. However, a dialogue strategy may not need to know the whole world knowledge in each state. It may also not need the whole action set per state. This research tackles such issues with **hierarchical sequential decision making**, which aims for a hierarchy of solutions (see Figure 1.3). Under this approach dialogue states can be described at different levels of granularity, where actions can execute behaviour with either dialogue acts or sub-dialogues. This approach offers the following benefits. First, modularity helps to solve sub-problems that may be easier to solve than the whole problem. Second, sub-problems may include only relevant dialogue knowledge in the states and relevant actions, thus reducing significantly the size of possible solutions: consequently they can be found faster. Last, sub-solutions can be reused when dealing with new problems. These benefits are possible at the cost of sub-optimal solutions. Nonetheless, they may be well worth the gains in terms of scalability to large systems. This thesis describes how to apply this approach to dialogue strategy learning.

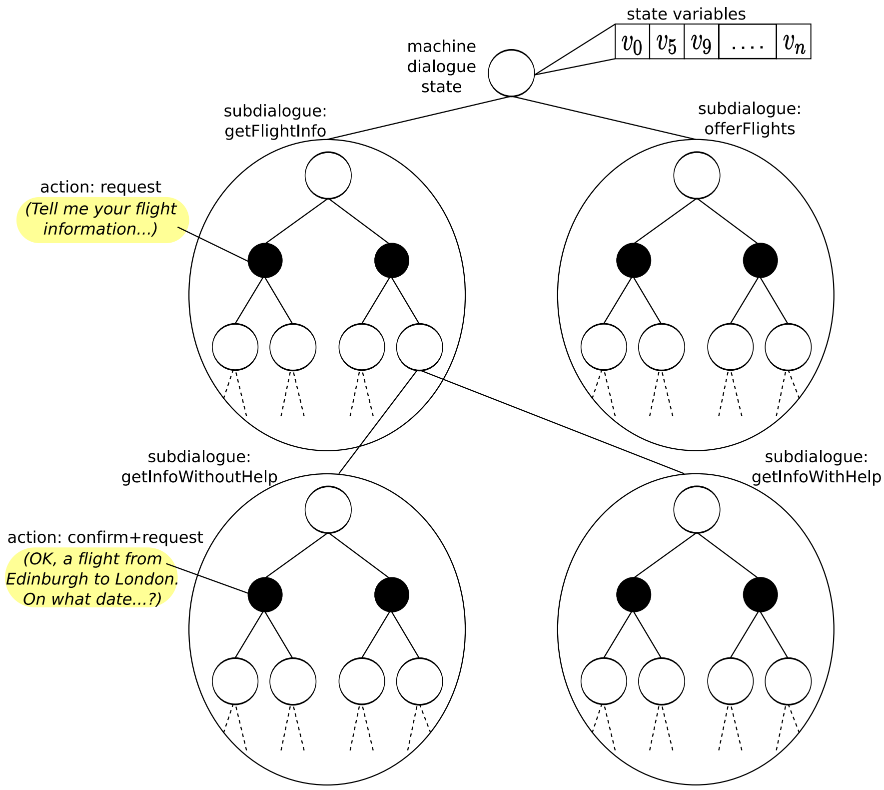


Figure 1.3: *Illustration of hierarchical sequential decision-making for spoken dialogue, where empty circles represent dialogue states with knowledge at different levels of granularity, and their transitions result from executed high- and low-level actions.*

1.4 Contributions

The following contributions are derived from the work described in this thesis:

(1) The Semi-Markov Decision Process (SMDP) model for spoken dialogue. This research proposed the SMDP model for dialogue strategy learning. Other models from previous investigations mostly use flat methods corresponding to the left branch of Figure 1.4. This contribution proposed a ‘divide and conquer’ approach using a hierarchy of SMDPs, where every SMDP represents a sub-dialogue in the conversation. This approach produced dramatic state-action space reductions of more than 99%, showed itself to be feasible for a spoken dialogue system with a flat state-action space of 10^{23} state-actions, and is promising for larger-scale systems.

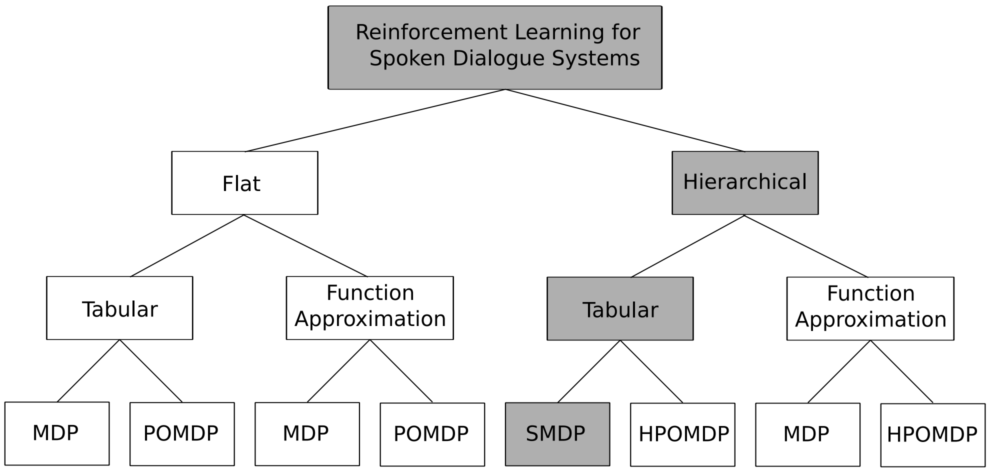


Figure 1.4: *Taxonomy of stochastic sequential decision-making for spoken dialogue. The shaded branch shows the model that forms the principal focus of this thesis.*

(2) Partially specified dialogue strategies. This concept puts together hand-crafted dialogue behaviours with learnt ones into a single framework. The former consist of hierarchical finite state machines using deterministic state transitions for actions easy to specify and stochastic state transitions for actions less easy to specify. The latter are designed by a hierarchical reinforcement learning agent. This contribution includes a learning algorithm called ‘HAM+HSMQ-Learning’ that combines two existing algorithms in the literature of hierarchical reinforcement learning.

(3) Evaluation of learnt dialogue behaviours with real users. This includes the development of a spoken dialogue system, with two metrics to evaluate simulated user behaviour, and a metric for evaluating baseline dialogue strategies. The generated real dialogues were crucial to evaluate fully-learnt, semi-learnt and baseline machine dialogue behaviours; and also to evaluate the realism of simulated dialogues.

1.5 Outline

The rest of this thesis is structured as follows:

- Chapter 2 presents a survey that bridges the fields of reinforcement learning and spoken dialogue systems. This chapter reviews some of the previously proposed approaches for learning spoken dialogue strategies and also reviews approaches for simulating and evaluating human-machine dialogues.

- Chapter 3 surveys hierarchical reinforcement learning methods, and focuses on approaches based on the Semi-Markov Decision Process (SMDP) model. It also discusses methods with more potential application to spoken dialogue.

- Chapter 4 proposes a simulation framework for generating human-machine conversations at the dialogue-act level using a heuristic approach. This chapter also describes metrics for evaluating user simulations, and a baseline dialogue strategy for assessing learnt dialogue behaviours.

- Chapter 5 proposes an approach for learning dialogue strategies using a hierarchy of Semi-Markov decision processes and hierarchical reinforcement learning. Experiments were performed in the flight booking and travel planning domains.

- Chapter 6 extends the dialogue optimization approach of the previous chapter with the concept of 'partially specified dialogue strategies'. Such strategies combine prior expert knowledge and learnt behaviour into a single framework.

- Chapter 7 evaluates the performance of a travel-planning spoken dialogue system with three behaviours: deterministic, fully-learnt and semi-learnt. This is the largest dialogue system using the reinforcement learning paradigm so far investigated in the literature. In addition, it evaluates simulated user behaviour based on data from real dialogues.

- In chapter 8 the thesis is summarized, promising future directions are discussed and the findings on hierarchical dialogue strategy learning are listed.

Three appendices complement the chapters above as follows: first, appendix A lists the notations used for reinforcement learning dialogue agents. Second, appendix B describes dialogue data structures used to represent the knowledge of both conversants. Finally, appendix C is a sample real dialogue showing hierarchical states, hierarchical actions, and corresponding machine and user utterances.

Chapter 2

Reinforcement learning for spoken dialogue systems

This chapter reviews literature in the field of reinforcement learning applied to spoken dialogue systems. It describes the proposal of 'dialogue as an optimization problem' (Levin and Pieraccini, 1997), which aims to contribute towards the development of more sophisticated dialogue systems. Section 2.2 reviews the tabular reinforcement learning framework. Section 2.3 describes previous work on reinforcement learning for dialogue strategy design. Section 2.4 describes previous work on dialogue simulation, aiding the facilitation of the task of learning dialogue strategies. Section 2.5 discusses some of the current challenges of learning efficient and effective dialogue behaviours for spoken dialogue systems. The last section summarizes the key points of the chapter.

2.1 Dialogue as an optimization problem

Dialogue strategies control the behaviour of spoken dialogue systems, and have been mainly hand-crafted by system designers and developers. Several approaches have been proposed for such a purpose: finite state, frame-based, agenda-based, information state, plan-based, and agent-based. The finite state approach is the simplest, and is suitable for system-initiative interactions, where the user answers questions in the form of simple commands (McTear, 1998). The frame-based and agenda-based approaches are suitable for mixed-initiative interactions, where the user can provide several items of information in any order (Goddeau et al., 1996; Chu-Carroll, 1999; Rudnicky and Wu, 1999; Seneff and Polifroni, 2000; Pieraccini et al., 2001; Bohus and Rudnicky, 2003). The information state approach is also suitable for mixed-initiative interactions, where an action is triggered from a set of rules and a given dialogue state (Larsson and Traum, 2000). The plan-based approach is suitable for collaborative dialogues (Rich and Sidner, 1998). The agent-based approach is suitable for complex dialogue behaviour, also includes planning, and involves behaviour in dynamically changing environments (Allen et al., 2001b,a). However, none of these approaches automate or optimize the dialogue strategy design. They usually require lengthy cycles of refinement in order to fully deploy dialogue systems with reasonable performance.

Levin and Pieraccini (1997) cleverly observed that there were no scientific guiding principles for designing the behaviour of spoken dialogue systems, which suggests that this task can be considered more as an art, rather than engineering or science. This issue motivated them to cast the problem of dialogue strategy design as an *optimization problem*. This proposal indeed matches the directions of *intelligent agents*, where they have to behave rationally by choosing the best actions according to some performance measure (Russell and Norvig, 2003). In this context, automating the dialogue strategy design shifts the practice from hand-coded static behaviours to automatic and adaptive behaviours.

The idea of **dialogue as an optimization problem** is as follows: given a set of dialogue states, a set of actions, and an objective cost function, an optimal dialogue strategy minimizes the objective function by choosing the actions leading to the lowest cost for every reached dialogue state. Such states describe the system's knowledge about the conversation (e.g. user input, database information, user information, etc.). The action set describes the system's capabilities (e.g. asking or confirming information, querying a database, giving help, etc.). The cost function assigns a cost for each taken action. In this way, a dialogue can be seen as a finite sequence of states, actions and costs $\{s_0, a_0, c_1, s_1, a_1, ..., c_{t-1}, s_t\}$, where the goal is to find an optimal strategy automatically. Levin and Pieraccini (1997); Levin et al. (1998, 2000) also suggested employing the reinforcement learning framework for such a task. But, optimizing dialogue strategies is not a simple process, specially for large spoken dialogue systems.

This chapter presents a survey bridging the fields of *reinforcement learning* and *spoken dialogue systems*. It introduces the reinforcement learning framework and describes approaches for optimizing dialogue strategies. It also surveys recent advances in the related field of dialogue simulation. Finally, this chapter discusses issues that currently limit the practical application of reinforcement learning dialogue systems. In this survey it was found that most of the literature has ignored the hierarchical learning approach, and therefore it has been identified as a significant research omission.

2.2 Background on reinforcement learning

Reinforcement learning is a computational approach to building agents that learn their behaviour by interacting with an environment (Kaelbling et al., 1996; Sutton and Barto, 1998; Bertsekas and Tsitsiklis, 1996). A **reinforcement learning agent** senses and acts in its environment in order to learn to choose optimal actions to achieve its goal. It is not given a form of teacher, like other machine learning approaches such as supervised learning that learn from examples (Russell and Norvig, 2003; Mitchell, 2004). Instead, it has to discover by trial-and-error search how to act in a given environment. For example, a robot may have sensors such as cameras and sonars to perceive the environment state, and actions that change its state such as moving in different directions. For each action the agent receives feedback (also referred to as a reward or reinforcement) to distinguish what is good and what is bad. The agent's task is to learn a policy or control strategy for choosing the best actions in the long run that achieve its goal. For such a purpose the agent maintains a **cumulative reward** for each state or state-action pair.

More specifically, reinforcement learning systems have four main elements: a policy, a reward function, a value function, and optionally, a model of the environment. A **policy** defines the behaviour of the learning agent. It consists of a mapping from states to actions – for each state the agent chooses the action with the highest learnt value. A policy can be represented with a look-up table, neural network, decision tree, or with a search algorithm. Policies are the core of reinforcement learning systems because they are sufficient to determine the agent's way of behaving. A **reward function** specifies how good the chosen actions are. It maps each perceived state-action pair to a single numerical reward. The reward function awards the agent for its good or bad actions, but only awards immediate actions. The ultimate objective of a learning agent is to maximize the cumulative reward it receives in the long run, from the current state and all subsequent next states. A **value function** specifies what is good in the long run. The value of a given state is the total reward accumulated in the future, starting from that state. The learning agent's action-selection mechanism will be based on actions with the highest values, not with the highest rewards. The efficient estimation of values is arguably the most important component of reinforcement learning algorithms. Finally, the **model of the environment** is something that mimics the environment's behaviour. A simulated model of the environment may predict the next environment state from the current state and action. Reinforcement learning algorithms using such a model perform 'model-based learning', otherwise they perform 'model-free learning'. The environment is usually represented as a Markov Decision Process (MDP) or as a Partially Observable MDP (POMDP).

Reinforcement learning is distinguished from other machine learning approaches by the following characteristics: **trial-and-error search** and **delayed reward**. In the former the agent has to try all actions per state many times in order to discover which actions lead to the highest cumulative reward. In the latter, the executed actions affect not only the current reward, but also the subsequent rewards. In many problems such as games the reward is only given at the end (Tesauro, 1995), which has to be back-propagated accordingly to the actions that produced such reward. In summary, reinforcement learning agents employ their own experience in order to improve their performance over time.

2.2.1 Markov decision processes

A reinforcement learning agent interacts with an environment that can be described by a Markov Decision Process (MDP) – see Figure 2.1. An MDP is a mathematical model used to optimize stochastic sequential decision making problems (Putterman, 1994; Sutton and Barto, 1998). This model is defined as a 4-tuple $<S, A, T, R>$ characterized as follows:

- S, is a set of states in the environment, where $S = \{s_0, s_1, ..., s_N\}$ and s_t is the state at time t. The states in an MDP are directly observable, used to describe all different situations in the environment, and the basis for action-selection. In an episodic task, the state set includes non-terminal states and terminal state (s). The state at time s_{t+1} is also denoted as s'.

- A, is the set of actions available in the environment, where $A = \{a_0, a_1, ..., a_M\}$ and a_t is the action at time t. When action a_t is executed it changes the current state of the world from s_t to s_{t+1}. The action at time a_{t+1} is also denoted as a'.

- $T(s', a, s)$, is a state transition function that observes the next state s' given the current state s and action a. The state transitions are represented with a conditional probability distribution $P(s'|s, a)$ satisfying $\sum_{s' \in S} P(s'|s, a) = 1, \forall(s, a)$.

- $R(s'|s, a)$, is the reward function that specifies the immediate reward r_t at time t given to the agent for choosing action a when the environment makes a transition from s to s'. The reward at time r_{t+1} is also denoted as r'.

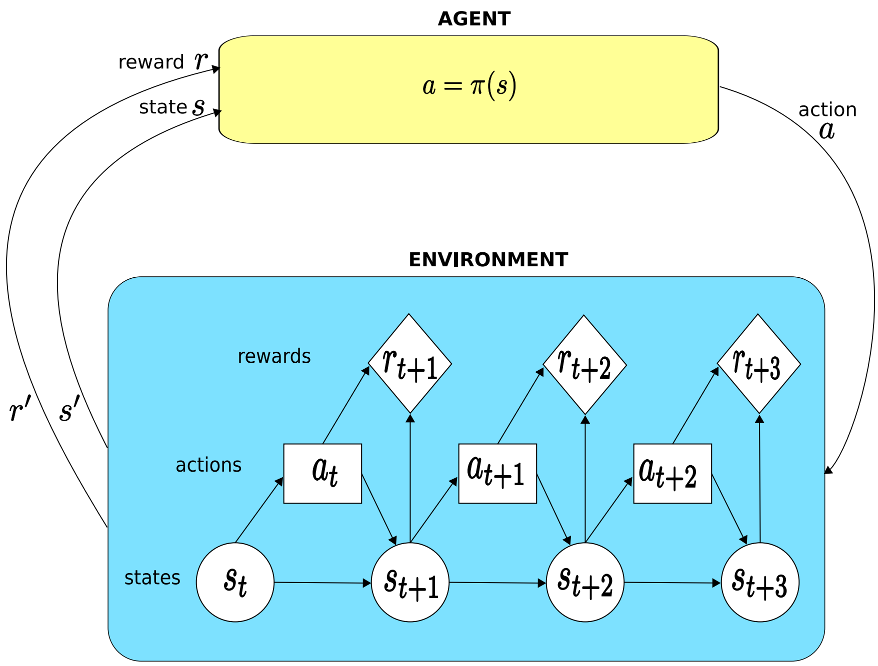


Figure 2.1: *The agent-environment interaction for MDP-based reinforcement learning.*

The solution to a Markov decision process is a decision-making function or policy π, which is a mapping from environment states $s \in S$ to actions $a \in A$ with probability $\pi(s,a)$. The optimal solution for an MDP is that of taking the best action a_t available in state s_t, i.e. the action that collected as much reward as possible over time. A given sequence of states, actions, and rewards $\{s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2 \ldots\}$, receives a total cumulative discounted reward expressed as

$$r = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots \gamma^{\tau-1} r_\tau = \sum_{k=0}^{\tau-1} \gamma^k r_{k+1}, \tag{2.1}$$

where the discount rate $0 \leq \gamma \leq 1$ makes future rewards less valuable than immediate rewards as it approaches 0. Such sequences can be episodic or continuing. The former last a finite number of time steps τ. The latter last an infinite number of time steps $\tau = \infty$ and the rewards must be discounted with $\gamma < 1$. In the equation above, the term on the right-hand side is referred to as 'the expected value of the reward', and can be computed recursively using a state-value function $V^\pi(s)$, which returns the value of starting in state s and then following policy π thereafter. The value-function is defined by the Bellman equation for V^π expressed as

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P(s'|s,a) \left[R(s'|s,a) + \gamma V^\pi(s') \right]. \tag{2.2}$$

Alternatively, the expected value of the reward can be also computed recursively using an action-value function $Q^\pi(s,a)$, which returns the cumulative reward of starting

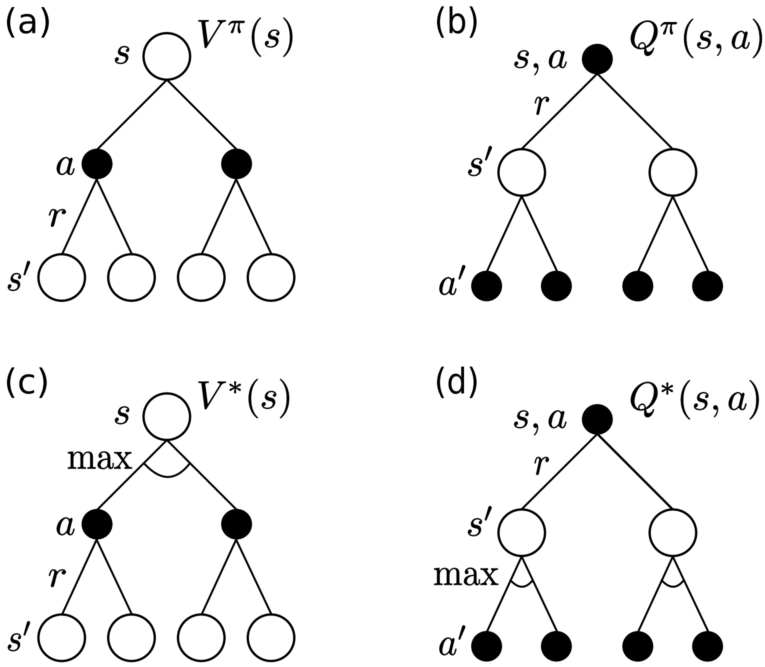


Figure 2.2: *Backup diagrams for (a) state value function, (b) action-value function, (c,d) optimal state- and action-value functions, respectively* (Sutton and Barto, 1998).

in state s, taking action a and then following policy π thereafter. The action-value function is defined by the Bellman equation for Q^{π} expressed as

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)\left[R(s'|s,a) + \gamma V^{\pi}(s')\right]. \tag{2.3}$$

The Bellman equations for V^{π} and Q^{π} are illustrated in Figures 2.2(a) and 2.2(b). They show the relationships when value information is carried back to the current state (or state-action pair) from the next states (or state-action pairs), these operations are therefore referred to as *backups*. An optimal policy π^{*} can be found by using the following Bellman equations that represent a system of equations, one for each state:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) = \max_{a} \sum_{s'} P(s'|s,a)\left[R(s'|s,a) + \gamma V^{*}(s')\right], \tag{2.4}$$

or state-action pair:

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)\left[R(s'|s,a) + \gamma \max_{a'} Q^{*}(s',a')\right]. \tag{2.5}$$

Figures 2.2(c) and 2.2(d) show the backups for the optimal functions V^{*} and Q^{*}. Finally, an **optimal policy** performs action-selection according to

$$\pi^{*}(s) = \arg\max_{a} Q^{*}(s,a). \tag{2.6}$$

The optimal policy can be learnt by either classical dynamic programming methods such as value iteration (Putterman, 1994), or by reinforcement learning methods such as Q-Learning or SARSA (Kaelbling et al., 1996; Bertsekas and Tsitsiklis, 1996). The next subsection explains why the latter are preferred.

2.2.2 Tabular reinforcement learning algorithms

A reinforcement learning algorithm has the objective of computing an optimal policy for behaving in a given environment described by a Markov decision process. A learning algorithm computes a value function V^* or action-value function Q^* from the following dynamics: at each time step t, the algorithm is given the current environment state $s \in S$ and a set of actions $A(s) \in A$, the algorithm takes an action a and the MDP executes it, then the algorithm receives next state $s' \in S$ and reward r'. If the current state is a terminal state, the episode terminates its execution. This process is executed an infinite number of times until the learnt value function stabilizes.

Reinforcement learning algorithms offer two important advantages over classical dynamic programming: they are online and can employ function approximation to represent their knowledge. In the former, they do not require a full model of the environment (complete probability distributions of all transitions). In the latter, alternative representations can be used other than look-up tables. Figure 2.3 illustrates a unified view of reinforcement learning methods: Dynamic Programming (DP), Monte Carlo (MC) methods, and Temporal Difference (TD) learning (Kaelbling et al., 1996; Sutton and Barto, 1998). All of them are based on delayed rewards and can be distinguished in the way they employ backups: *sample* backups are based on a sample trajectory, *full* backups are based on all possible trajectories, *shallow* backups are based on a one-step trajectory, and *deep* backups are based on trajectories reaching a terminal state. In this way, DP employs full and shallow backups, MC employs sample and deep backups, and TD employs sample and shallow backups. Whilst DP requires complete knowledge of the environment, MC methods require only experience, namely sample sequences of states, actions and rewards. However, MC methods are not suited for step-by-step incremental computation. Furthermore, TD learning is a combination of DP and MC methods because it does not require a complete model of the environment and because it employs shallow backups. Each reinforcement learning method has its own strengths and weaknesses, and one may pick one over another depending on the task. It is perfectly reasonable to apply a joint method with aspects of more than one

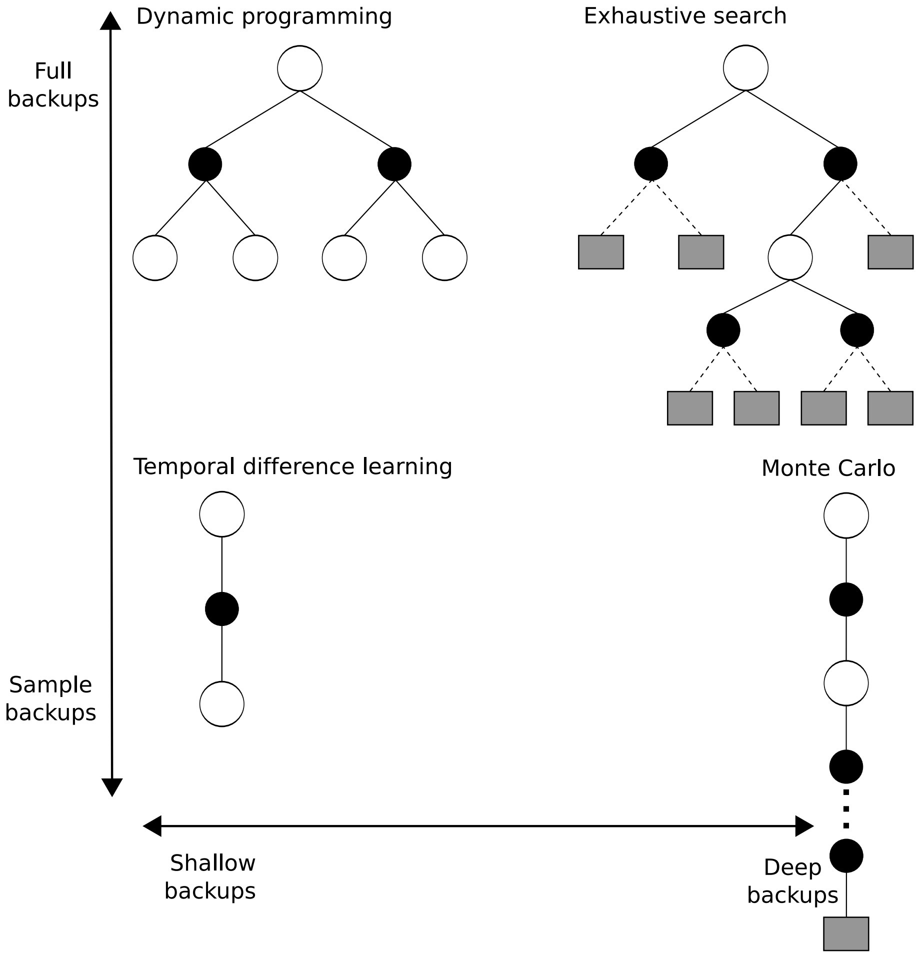


Figure 2.3: *A unified view of reinforcement learning methods* (Sutton and Barto, 1998), *they can be classified according to their type of backups. Notation: empty circles represent states, dark circles represent actions, and rectangles represent terminal states.*

kind, but these choices can be made later when the methods are used rather than when they are designed (Sutton and Barto, 1998).

One of the challenges in reinforcement learning is the trade-off between exploration and exploitation. The agent has to perform **exploration** in order to discover better behaviours, but it also has to perform **exploitation** of the already learnt behaviour in order to obtain more reward. In this dilemma, a learning agent must try different actions and progressively prefer those that seem to be the best. The basic methods for action-selection in reinforcement learning are ε-*greedy* and *softmax* (Sutton and Barto, 1998). In the former the agent performs exploitation with a fixed probability $1-\varepsilon$, and with probability ε performs exploration:

$$\pi(s) = \begin{cases} \arg\max_a Q(s,a) & \text{if} \quad p(random) \leq 1-\varepsilon \\ random(a \in A) & \text{otherwise.} \end{cases} \tag{2.7}$$

In the latter method the agent performs exploration-exploitation according to a probability distribution of cumulative rewards $Q(s,a)$:

$$P(a|s) = \frac{e^{Q(s,a)/T}}{\sum_{a' \in A} e^{Q(s,a')/T}}. \tag{2.8}$$

The parameter T represents the temperature used to decrease exploration over time.

One of the simplest and most popular reinforcement learning algorithms is **Q-Learning**, see algorithm 1 (Watkins, 1989). It computes Q-values according to

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]. \tag{2.9}$$

Q-Learning updates values for sample state-action pairs (s,a), where the execution of action a in state s yields state s' and reward r, γ is a discount rate in the range $[0,1]$, and α is a learning rate parameter that decays from 1 to 0; for example: $\alpha_t = 1/(1+v_t)$, where $v_t = visits_t(s,a)$ is the number of times that (s,a) has been visited until step t. Jaakkola et al. (1994) proved that if the learning agent has a finite state-action space, and if it tries every action infinitely often in every state, and if α is decayed according to

$$\lim_{T \to \infty} \sum_{t=1}^{T} \alpha_t = \infty \text{ and } \lim_{T \to \infty} \sum_{t=1}^{T} \alpha_t^2 < \infty, \tag{2.10}$$

then it converges to the optimal action-value function Q^* with probability 1.

A similar algorithm to Q-Learning called **SARSA** (State-Action-Reward-State-Action) computes the cumulative reward but without taking into account the optimal action in the next state s', and updates its values according to

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma Q(s',a') - Q(s,a) \right]. \tag{2.11}$$

These two algorithms differ in the way they approach the trade-off exploration and exploitation. The Q-Learning algorithm uses an *off-policy* approach because it performs learning based on two policies: a behaviour policy for exploration, and an estimation policy for exploitation. In contrast, the SARSA algorithm uses a single policy for both exploration and exploitation. The advantage of the former approach is that whilst the estimation policy behaves greedily, the behaviour policy samples all possible actions. This approach has received more attention for hierarchical learning (see chapter 3).

The reinforcement learning algorithms Q-Learning and SARSA have been extended in many different ways. For instance, they can incorporate eligibility traces to update all action-values per time step according to their eligibility[1], which may result in more efficient learning (Singh and Sutton, 1996; Sutton and Barto, 1998).

[1] An eligibility trace $e(s)$ is a value assigned for visiting state s, which gradually decays over time. Based on such eligibility traces, good or bad rewards in the future assign credit accordingly.

Algorithm 1 The Q-Learning algorithm

function Q-LEARNING(states S, actions A, transitions T, rewards R, discount γ)
    Initialize $Q(s,a)$ arbitrarily, and initialize α to 1
    repeat(for each episode):
        Initialize s
        repeat(for each step of episode):
            Choose a from s using policy derived from Q (e.g. ϵ-greedy)
            Take action a
            Observe r from R
            Observe s' from T
            Decay α (e.g. $\alpha = 1/(1+visits(s,a))$
            $Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
            $s \leftarrow s'$
        until s is terminal
    until convergence
    return $Q(s,a)$
end function

Note: For simplification purposes, the learning algorithms described in the rest of this thesis declare a more compact set of parameters and omit the update of the learning rate parameter α.

2.3 Approaches for dialogue optimization

Several approaches have been proposed for optimizing spoken dialogue strategies using reinforcement learning. This section describes the strengths and weaknesses of four approaches, where each one provides a novel optimization process.

2.3.1 Dialogue as a Markov decision process

Levin and Pieraccini (1997) proposed learning the behaviour of spoken dialogue strategies using the Markov Decision Process (MDP) formalism. A dialogue-based MDP is characterized by a finite set of dialogue states S, a finite set of actions A corresponding to dialogue acts, a state transition function $T(s,a,s') = P(s'|s,a)$, a reward function $R(s,a,s')$, and a dialogue strategy $a = \pi(s)$ mapping states to actions. The state transition function employs the *Markov property*, which specifies that the dialogue state at time $t+1$ depends only on the dialogue state and action at time t, rather than the full history of state and actions, expressed as

$$P(s_{t+1}|s_t, a_t, r_t, s_{t-1}, a_{t-1}, ..., r_1, s_0, a_0) = P(s_{t+1}|s_t, a_t). \quad (2.12)$$

A dialogue-based MDP is episodic because human-machine dialogues have a finite number of interactions, and differs from the standard formulation as follows: (i) probabilistic state transitions $P(s'|s, a)$ must generate dialogues that make sense to humans, alternatively, any state transition can be allowed on simulated environments; and (ii) the learnt dialogue policy $\pi^*(s)$ must perform action-selection with reasonable behaviour. Most of the previous work in the field has focused on the MDP model, and a list of representative investigations is shown in Table 2.1. It can be observed that most of them have focused on dialogue policies with few slots (semantic concepts), learnt in simulated environments, and few of them have been evaluated with real users.

Three main problems affect the practical application of the MDP model for dialogue strategy learning: the curse of dimensionality, partial observability, and learning from real interactions. In the first, the state space growth is exponential in the number of state variables (e.g. state representations with $\{10, 20, 30, 40, 50\}$ binary state variables yield $\{10^3, 10^6, 10^9, 10^{12}, 10^{15}\}$ unique states, respectively). In the second, the dialogue agent operates under uncertainty (the most obvious source is automatic speech recognition errors, but not the only source). In the third, reinforcement learning methods require many dialogues to find optimal policies. These problems offer motives for proposing alternative optimization approaches.

2.3.2 Dialogue as a partially observable MDP

Roy et al. (2000) proposed employing the Partially Observable Markov Decision Process (POMDP) model for robust spoken dialogue behaviour, which is a generalisation of the MDP model, but handles the uncertainty perceived from the environment. It is defined as a 6-tuple $<S, A, \Omega, T, O, R>$ characterized as follows: (1) S is a set of states, (2) A is a set of actions, (3) $\Omega = \{o_1, o_2, ..., o_n\}$ is a set of observations or perceptions from the environment (e.g. keywords from the user utterances), (4) $T(s, a, s')$ is a transition function for transitioning to the next state s' given the current state s and action a with probability $P(s'|s, a)$, (5) $O(s, a, o)$ is the observation function that the agent will perceive observation o from selecting action a in state s with probability $P(o|s, a)$, and (6) $R(s, a, s')$ is the reward function that specifies the reward given to the agent for choosing action a when the environment makes a transition from s to s'.

Table 2.1: *A summary of previous research on MDP-based dialogue strategy learning.*

Author(s)	Slots	States	Actions	Learning Algorithm	Real User Testing	Training Dialogues
(Levin et al., 1998, 2000)	5	111	12	MC1	No	Simulated
(Singh et al., 1999)	5	32	9	VI	No	Simulated
(Young, 2000)	2	36	5	VI, MC1	No	Simulated
(Litman et al., 2000; Singh et al., 2002)	3	42	2	VI	Yes	Real
(Goddeau and Pineau, 2000)	n	3^n	5	DP	No	Simulated
(Walker, 2000)	3	18	17	Q-Learning	Yes	Real
(Pietquin and Renals, 2002)	7	3^7	24	MC1	No	Simulated
(Scheffler and Young, 2002)	4	1229	6	$Q(\lambda)$	No	Simulated
(Denecke et al., 2004)	4	972	5	FVI, FA	Yes	Real
(Henderson et al., 2005; Lemon et al., 2006a)	4	10^{87}	70	SARSA(λ), LFA	No Yes	Real Simulated
(Frampton and Lemon, 2005, 2006, 2008)	4 4 3	1539 784 ?	6 7 ?	SARSA(λ) SARSA(λ) SARSA(λ)	No No Yes	Simulated Simulated Simulated
(English and Heeman, 2005)	4	2^5	5	MC2	No	Simulated
(Schatzmann et al., 2005b)	4	81	256	Q-Learning	No	Simulated
(Pietquin and Dutoit, 2006; Pietquin, 2007)	7 5	2187 32768	25 6	$Q(\lambda)$ $Q(\lambda)$	No No	Simulated Simulated
(Cuayáhuitl et al., 2006a)	20	4127	26	Q-Learning	No	Simulated
(Prommer et al., 2006)	3	16384	8	Watkins(λ)	No	Simulated

Abbreviations: MC1 = Monte Carlo with exploratory starts; MC2 = On-policy Monte Carlo; DP = Dynamic programming; VI = Value iteration; FVI = Fitted VI; FA = Function approximation; LFA = Linear function approximation.

Because environment states are partially known, the solution for a POMDP is a function mapping belief states to actions (Kaelbling et al., 1998). A belief state $b(s)$ is a probability distribution over S. Thus, a POMDP can be seen as an MDP over a belief space, where the observable states are replaced by belief states. When the agent takes action a and receives observation o, its belief on the next state s' is updated as:

$$b(s') = \frac{O(s', a, o) \sum_{s \in S} T(s, a, s') b(s)}{p(o|a, b)}. \tag{2.13}$$

The dynamics in a POMDP can be summarized in the following way: the agent executes action $a = \pi^*(b)$ from the current belief state b, receives observation o and reward r, computes the next belief state b' using equation 2.13 (this is called *belief monitoring*), and repeats the process until the end of the conversation.

Three main problems affect the practical application of POMDPs to spoken dialogue: the curse of dimensionality, the curse of history, and learning from real interactions. The first and the last problems were described in the previous approach. The curse of history refers to the number of distinct possible action-observation histories with the planning horizon (Pineau et al., 2006; Spaan and Vlassis, 2005). Pineau et al. (2001) optimized a hierarchy of POMDPs with a bottom-up approach, but still only suitable for small state-action spaces. Most previous research has focused on keeping belief monitoring tractable by using some sort of compression of the belief state (Roy et al., 2000; Zhang et al., 2001; Williams, 2006, 2007b,c; Atrash and Pineau, 2006; Young et al., 2007; Williams, 2007d; Thomson et al., 2008; Henderson and Lemon, 2008).

2.3.3 Dialogue control using function approximation

Most of the currently available reinforcement learning algorithms approximate the state-value function or action-value function using a look-up table. Although they work well for small state spaces, they quickly become intractable due to the curse of dimensionality problem. A solution for dealing with large state spaces is to use function approximation, which replaces the table with a function representation such as a linear function, decision tree, neural network, or kernel-based method, among others. Such a representation is an approximation because the true value function might not be represented in the chosen form. For example, in the weighted linear function

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \ldots + \theta_n f_n(s) \tag{2.14}$$

with set of features $f_i \in F$ and parameters $\theta = \{\theta_1, ...\theta_n\}$, a reinforcement learning agent can learn values for the parameters θ, where the utility function $\hat{U}_\theta$ approximates to the true utility function (Russell and Norvig, 2003).

Function approximation approaches represent value functions of very large state spaces in a practical way, but their main benefit is that they allow the learning agent to generalise from visited states to unseen states. This is possible because the updating of θ_i values also updates the value function, which then affects all states. But it also may lead to an unstable function approximation (Gordon, 2000). A Q-Learning agent can update the learning parameters using the following update rule between successive states, where $\hat{Q}$ approximates the utility function $\hat{U}$ of equation 2.14:

$$\theta_i = \theta_i + \alpha[r + \gamma \max_{a'} \hat{Q}_\theta(s', a') - \hat{Q}_\theta(s, a)] \frac{\partial \hat{Q}_\theta(s, a)}{\partial \theta_i}. \tag{2.15}$$

Previous investigations have employed function approximation to learn dialogue policies efficiently from small data sets. Denecke et al. (2004) proposed to employ two state-action spaces: abstract and concrete. The former includes all the state-action pairs and the latter includes only the most frequently visited. They performed learning on the concrete state-action space and generalise the learnt values to the abstract space. Henderson et al. (2005) proposed a hybrid approach for very-large state spaces, where reinforcement learning is used to optimize a measure of dialogue reward and supervised learning is used to restrict the learnt policy to the portion of existing data. In addition, Rieser and Lemon (2007) applied an approach based on hierarchical reactive planning, SARSA, and linear function approximation (Shapiro and Langley, 2002). These investigations did not take into account uncertainty in the conversation, and the convergence to an approximated optimal solution is more difficult to guarantee.

2.3.4 Dialogue control using evolutionary reinforcement learning

Evolutionary Algorithms (EAs) are used for stochastic search problems inspired by the theory of evolution and natural selection. They operate in policy space rather than value-function space (Moriarty et al., 1999). The goal in this approach is to search for a policy or solution that is progressively refined $\{\pi, \pi', \pi'', ..., \pi^*\}$ until finding the optimal policy π^*. Policies are encoded into structures called ‘chromosomes’. In table-based policy representations they consist of condition-action rules, where each condition is a predicate that represents a set of states. A fitness function or performance measure is used for ranking potential solutions. EAs operate on an initial population of chromosomes and iterate as follows: (1) evaluate the fitness of chromosomes, (2) select parent chromosomes stochastically according to their fitness, (3) evolve parent chromosomes, and (4) replace the old population with the evolved parents. This procedure of finding the best solution is also referred to as 'survival of the fittest'.

Previous work in dialogue strategy optimization has applied the eXtended Classifier System (XCS) model (Toney et al., 2006b,a; Toney, 2007), a generalisation of the Learning Classifier System (LCS) model. In this model, condition-action rules are represented with strings based on the symbols $\{0,1,\#\}$: e.g. the condition string 1#1 encapsulates the states 101 and 111. This model belongs to a class of evolutionary reinforcement learning methods, where a genetic algorithm is used to evolve and evaluate a population of rules, and a reinforcement learning algorithm is used to assign rewards to the rules. The XCS learning algorithm computes cumulative rewards in a similar way to Q-Learning. This approach was investigated on a flight booking spoken dialogue system with 10^9 unique state-actions. In general, this approach does not properly address partial observability. It mitigates the curse of dimensionality problem by using a more compact representation with regions of state-actions, but less optimal solutions may be found than tabular value functions. This approach can also be combined with function approximation approaches to solve larger problems (Whiteson and Stone, 2006).

2.3.5 Learning with real and simulated dialogues

Dialogue strategy learning has been examined using two different conversational environments: simulated and real. Each environment has its own strengths and weaknesses. In both cases learning has been performed offline, and not during the course of the dialogue with real users. Similarly, dialogue strategy testing has been examined with both environments, and the simulated ones have been preferred due to the extensive resources required by the real environments.

On the one hand, since Levin et al. (2000) coined the term 'dialogue as an optimization problem', they observed that a large number of dialogues would be required for such a purpose. This motivated them to employ **simulated dialogues**. They proposed to use supervised learning for training a probabilistic model of user behaviour, and to use reinforcement learning for optimizing the dialogue strategy. The strength of this approach is that dialogue simulators quickly generate a large number of dialogues. The main benefit of this approach is in the practical application because it can generate infinite amounts of dialogues. Its drawback is that simulated behaviour may be different from the real behaviour. Nonetheless, this approach has been widely used in most of the previous work in the field (Levin et al., 2000; Young, 2000; Goddeau and Pineau, 2000; Scheffler and Young, 2002; Lemon et al., 2006a; Frampton and Lemon, 2005, 2006; English and Heeman, 2005; Schatzmann et al., 2005b; Pietquin and Dutoit, 2006; Williams, 2006; Pietquin, 2007; Cuayáhuitl et al., 2006a; Prommer et al., 2006; Toney, 2007; Rieser and Lemon, 2007; Young et al., 2007; Thomson et al., 2008), among others. This approach was extended by Goddeau and Pineau (2000); Pietquin and Renals (2002) in order to learn dialogue policies in the presence of speech recognition errors. In addition, Schatzmann et al. (2005b) found that the quality of the learnt dialogue strategies is strongly dependent on the simulated user model, where good user models help to find better policies than poor user models. This suggests that dialogue strategy learning should employ realistic simulated user behaviour.

On the other hand, learning dialogue strategies using **real dialogues** is very appealing because it employs the dynamics from real conversational environments. Related work using real dialogues has adopted an offline learning approach, where researchers collect exploratory dialogue data from a real system and then use it to learn dialogue behaviours (Singh et al., 1999; Walker, 2000; Singh et al., 2002; Denecke et al., 2004).

Litman et al. (2000) proposed the following methodology for optimizing dialogue strategies on small state-action spaces: (a) design an appropriate reward function, state representation, and hand-coded state-action space – mapping states to reasonable actions; (b) build an initial state-based training system and deploy it to collect exploratory data; (c) use the collected data to build an empirical MDP; (d) compute the optimal dialogue policy; and (e) redeploy the system using the learnt state-action mapping. Although this methodology was applied successfully, it might not be very practical because larger state-action spaces are usually needed, and because the currently available methods for dialogue strategy learning usually require a large number of dialogues.

Walker (2000) extended the previous methodology by estimating the reward function (instead of handcrafting it) using the PARADISE framework (Walker et al., 1997), based on the metrics shown in Table 2.2 and a data set of exploratory dialogues. The performance function is estimated with a multivariate linear regression: it employs user satisfaction as the dependent variable; and task success, dialogue quality and dialogue efficiency as independent variables. The performance for any dialogue is defined by

$$Performance = (\alpha * N(\kappa)) - \sum_{i=1}^{n} w_i * N(c_i), \qquad (2.16)$$

Table 2.2: *Evaluation metrics for spoken dialogues (Walker, 2000).*

Group	Metrics
Dialogue efficiency	Elapsed time, system turns, user turns
Dialogue quality	Mean recognition score, time-outs, rejections, helps, cancels, barge-ins
Task success	Task success as per survey
User satisfaction	The sum of TTS performance, ASR performance, task easy, interaction pace, user expertise, system response, expected behaviour, comparable interface, future use

where α is a weight on task success (κ), c_i are the cost functions of efficiency and qualitative metrics weighted by w_i, and N is a normalization function. The estimated performance function can be tested using cross-validation on training and test data sets. If both data sets are statistically indistinguishable, then it can be assumed that the performance function will generalize to unseen dialogues.

In summary, real dialogues can be used if the state-action space is small enough to be sufficiently explored, or if the reinforcement learning algorithms are very efficient using dialogues that make sense to real users. In contrast, simulated dialogues can be used if the state-action space cannot be sufficiently explored by real users. But, they should be as realistic as possible in order to optimize good quality dialogue strategies.

2.3.6 Evaluation of learnt dialogue policies with real users

Previous work in evaluating learnt dialogue policies with real users has reported results based on average reward and the metrics shown in Table 2.2. Such evaluations used different types of baseline behaviours. Firstly, compact state spaces with reasonable actions were used to generate exploratory dialogues (Litman et al., 2000; Walker, 2000; Denecke et al., 2004). Secondly, hand-coded dialogue strategies were used to specify deterministic behaviour (Lemon et al., 2006b; Toney, 2007; Frampton and Lemon, 2008). Thirdly, alternative models for dialogue control such as Rieser and Lemon (2008) used policies based on decision trees to evaluate MDP-based policies, and Gasic et al. (2008) used MDP-based policies to evaluate POMDP-based policies.

In general, learnt dialogue policies usually outperform the given baseline behaviour. However, most of the evaluations do not demonstrate how good the baselines are, with some exceptions (Singh et al., 2002). A learnt dialogue strategy could easily outperform a poor baseline, but may find difficulties in outperforming better baselines. Baseline dialogue strategies should be measured to find if the learnt policies are better than state-of-the-art behaviours. Establishing standardised baselines would contribute towards better benchmarks, but they remain to be established.

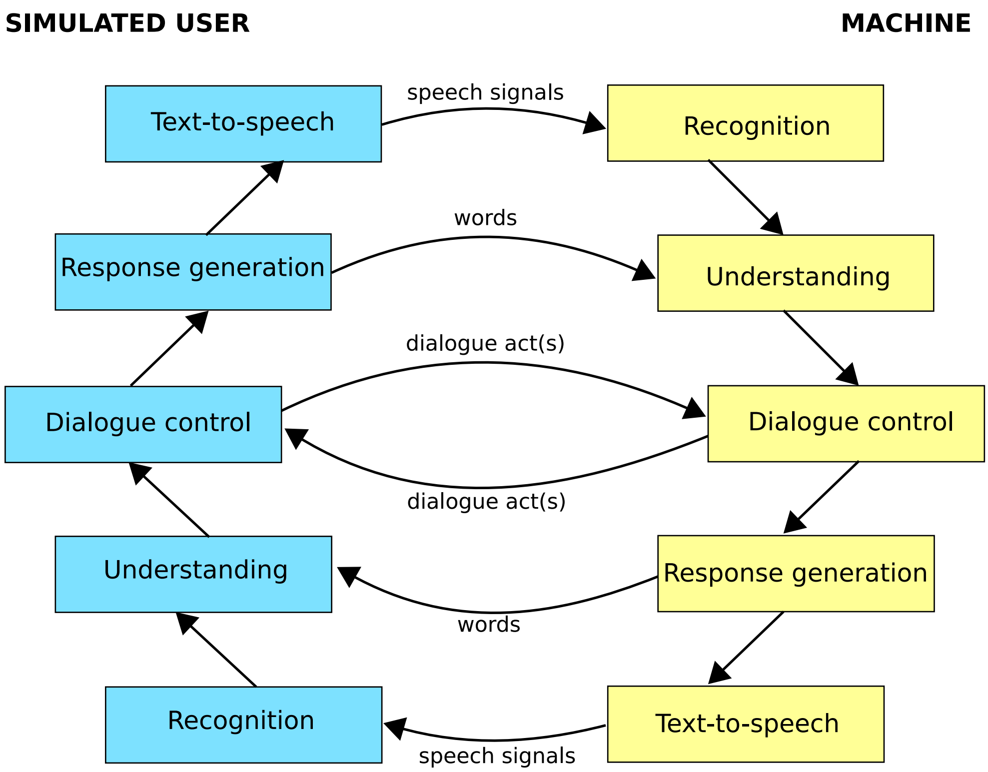


Figure 2.4: *Conversational interaction between a simulated user model and a spoken dialogue system* (*machine*)*, adapted from Eckert et al. (1997).*

2.4 Approaches for dialogue simulation

The simulation of human-machine task-oriented dialogues involves generating artificial dialogues between a spoken dialogue system and a simulated user (see Figure 2.4). The communication of both conversants can be achieved at different levels of granularity such as speech signals, words, and dialogue acts. The artificial data can be used to (re) train the machine's components. For example, words can be used to train language models, and dialogue acts can be used to train dialogue models. The latter have been widely adopted for reinforcement learning of dialogue strategies because conversations at the dialogue act level are useful for improving dialogue behaviours.

Generally speaking, *the problem addressed in user simulation is to predict the next realistic user response given a current approximate user dialogue state*. This is not a trivial task due to the fact that the user dialogue state space may be large. A user dialogue state can be represented with information such as the last machine dialogue act, the slot-values of all slots, the status of all slots, and so forth. Table 2.3 shows a summary of user simulation approaches[2]. They differ in two main aspects: (1) the way in which they represent the dialogue state, and (2) the way in which they choose user responses. The rest of this section categorises them into four broad approaches: rule-based, probabilistic, probabilistic-goal-directed, and deterministic-probabilistic.

Table 2.3: *Previous works on user simulation approaches for slot filling applications.*

Author(s)	Approach	Communication Level	Data Driven
(Eckert et al., 1997)	Bigram	Dialogue act	Partially
(Levin et al., 2000)	Constrained bigram	Dialogue act	Partially
(Scheffler, 2002)	Goal directed model	Dialogue act	Partially
(López-Cózar et al., 2003)	Rule-based	Words, speech	Hand-crafted
(Chung, 2004)	Rule-based	Words, speech	Hand-crafted
(Pietquin, 2004)	Goal directed model, BNs	Dialogue act	Hand-crafted
(Filisko and Seneff, 2005)	Rule-based	Words	Hand-crafted
(Filisko and Seneff, 2006)	Rule-based	Dialogue act	Yes
(Georgila et al., 2005a, 2006)	(Advanced) N-grams, LFC	Dialogue act	Yes
(Cuayáhuitl et al., 2005)	Hidden Markov models	Dialogue act	Yes
(Rieser and Lemon, 2006a)	Cluster-based model	Dialogue act	Yes
(Schatzmann et al., 2007a)	Agenda	Dialogue act	Hand-crafted
(Schatzmann et al., 2007c)	Hidden agenda	Dialogue act	Yes

Abbreviations: BNs = Bayesian networks, LFC = Linear feature combination.

2.4.1 Rule-based simulated user models

The behaviour of this approach is based on a set of rules that dictate how to act. Like any other simulated user model, their behaviour can be inferred from data, or specified in a heuristic way according to the system developer's experience. The former has the

[2] See Schatzmann et al. (2006) for a more detailed review on user simulation for dialogue systems.

advantage that more realistic behaviour can be generated, but its disadvantage is the cost of collecting and annotating the data (Filisko and Seneff, 2006). The latter has the advantage that it can be developed and modified without requiring annotated corpora, but its disadvantage is that the simulated behaviour may be significantly different from the real one. Previous work has employed heuristic behaviours to generate user responses at the speech and word levels of granularity (López-Cózar et al., 2003, 2008; Chung, 2004; Filisko and Seneff, 2005). They have been used to find problematic interactions, and to test the performance of spoken dialogue systems.

2.4.2 Probabilistic simulated user models

The behaviour of this approach is driven by conditional probability distributions for user dialogue act selection, and can also be hand-crafted or estimated from data. Eckert et al. (1997) proposed generating user responses based on a bigram model $P(u_t|s_t)$, where u_t is the user dialogue act at time t, and s_t is the last system's dialogue act. This model has been used in a number of investigations (Levin et al., 2000; Schatzmann et al., 2005a; Cuayáhuitl et al., 2006a; Hurtado et al., 2007; Williams, 2007a). Georgila et al. (2005a, 2006) extended the bigram model with n-grams varying from 2-grams to 5-grams. Georgila et al. (2006) also proposed simulations based on linear feature combination, mapping a vector of real-valued features $f(s)$ for the user dialogue state s to user actions a with probability $P(a|s)$. Cuayáhuitl et al. (2005) employed input-output HMMs (one per dialogue goal) to predict user dialogue acts with probability $P(u_t|q_t, s_t)$, and system dialogue acts with probability $P(s_t|q_t)$, where q_t are states in the HMMs. Rieser and Lemon (2006a) generated user responses driven by clusters that group together feature vectors based on their similarity.

These models are appealing because they can explore vast combinations of user responses using dynamics estimated from real dialogues. Their weakness is that they may generate incoherent behaviour – due to the smoothing of probability distributions to allow unseen user responses, or simply not applying enough constraints to responses.

2.4.3 Probabilistic-goal-directed simulated user models

The behaviour of this approach is based on consistent user responses following a *user goal*, aiming to mitigate the inconsistencies observed from purely random behaviour. Scheffler and Young (2000, 2001) proposed user dialogue behaviour based on a probabilistic finite state machine and a predefined user goal g, the latter being a data structure of slot-value pairs for the current dialogue. Pietquin (2004) combined the bigram model with a user goal to generate responses with probability $P(u_t|s_t,g)$, and extended this combined model with Bayesian networks (Pietquin and Dutoit, 2006). Schatzmann et al. (2007a) represented user dialogue states with an agenda and user goal, and generated responses from a prioritised stack of user dialogue acts. Schatzmann et al. (2007c) extended the agenda model with hidden user dialogue states, and trained this model from real data. In general, a user goal can be viewed as some sort of *knowledge base* for the simulated user model, where the more information it incorporates, the more helpful it is to generate more consistent user responses.

2.4.4 Deterministic-probabilistic simulated user models

The behaviour of this approach is a combination of the previous models because the user behaviour may be driven by rules, may incorporate probabilistic behaviour, and user responses can be constrained with a user goal or knowledge base. This combination aims to bring together the benefits of the approaches above. Consequently, the action selection mechanism for this approach can be fully hand-crafted (Toney, 2007; Cuayáhuitl et al., 2006b), fully-learnt from data, or a combination of both (Scheffler and Young, 2001; Scheffler, 2002; Torres et al., 2008). Whilst the former behaviour is suitable when dialogue data does not exist, the latter two behaviours are more suitable when dialogue data does exist for (re) training the simulated user model.

2.4.5 Evaluation of simulated dialogues

The evaluation of simulated user models has the purpose of assessing their quality in order to use the best models for dialogue strategy learning or testing. The overall goal is to find simulated user models that can help to build more sophisticated spoken dialogue behaviours. Previous work has proposed several evaluation metrics for simulations based on dialogue acts. They are summarized in Table 2.4, and can be grouped into the following approaches: dialogue similarity and system performance.

The **dialogue similarity** approach is based on the following assumption: given a set of metrics, a set of simulated dialogues, and a set of real dialogues – the realism of simulated dialogues increases as their scores approach those obtained by real ones. Most previously proposed evaluation metrics fall within this approach. Although there is no concrete definition for dialogue realism, researchers in the field agree that realistic simulated user behaviour must exhibit the property of *human-like behaviour*.

Table 2.4: *Evaluation metrics for human-machine dialogue simulation.*

Author(s)	Proposed metric
(Eckert et al., 1997)	Dialogue length, task success
(Schatzmann et al., 2005a)	Precision-recall, statistical metrics
(Schatzmann et al., 2005b)	Policy similarity
(Georgila et al., 2005a)	Perplexity
(Cuayáhuitl et al., 2005)	The Kulback-Leibler divergence
(Georgila et al., 2006)	Expected accuracy/precision/recall
(Rieser and Lemon, 2006a)	Pragmatic error rate
(Williams, 2007a)	The Cramér-Von Mises divergence

This property has been evaluated in different ways. Some investigations have used *dialogue length and success metrics* (such as average number of system turns, average number of dialogue acts, or binary task success per task) to give a rough indication of agreement between a set of real dialogues and a set of simulated ones (Eckert et al., 1997; Scheffler and Young, 2000, 2001; Scheffler, 2002; Schatzmann et al., 2005a; Filisko and Seneff, 2005, 2006; Cuayáhuitl et al., 2005). Other investigations have used *precision-recall and policy similarity metrics* to quantify how closely simulated dialogue acts resemble real ones (Schatzmann et al., 2005a,b; Georgila et al., 2005a; Cuayáhuitl et al., 2005; Rieser and Lemon, 2006a). Although these metrics evaluate how well a model can predict training and test data, they penalize highly simulated dialogues that do not occur in the real data. Other investigations have used *probabilistic metrics* to quantify the probabilistic similarity of simulated and real dialogues. Georgila et al. (2005a) proposed perplexity to evaluate how well a model predicts sequences of elements in a test data-set. Their assumption is that the lower perplexity the better. Cuayáhuitl et al. (2005) proposed the Kulback-Leibler divergence (distance) with discrete probability distributions of system/user dialogue acts. This metric assumes that the lower the divergence the better. In general, the metrics above are useful for giving a rough indication of the similarity between simulated and real dialogues. Their main weaknesses are that they are not suitable for properly penalizing unseen behaviour, and that they cannot distinguish if a given sequence of machine-user dialogue acts is realistic or not.

The **system performance** approach ranks simulated user models viewed as predictors of the performance of a dialogue system. Here, motivation derives from the fact that simulated user models should improve machine dialogue behaviours rather than generating human-like conversations. Williams (2007a) proposed computing the normalized Cramér-Von Mises divergence between real dialogue scores and simulated dialogue scores, where the scoring function is similar to a dialogue reward function. The assumption here is that as the predictive accuracy of the simulated user model increases, its divergence decreases: the lower the divergence, the better the simulated user model. Although this is a promising approach for evaluating user simulators, it is limited by the fact that it requires real dialogue data, which may not exist at early stages of system development. This limitation also applies to the metrics above that assume an existing dialogue data set.

2.5 Open questions in dialogue strategy optimization

To date, important advances have been made in the field of dialogue strategy optimization; however, the research questions described below currently remain open. This is by no means a complete list of research gaps, but it gives some idea of current problems to be tackled. Further investigations can take them into account in order to build spoken dialogue systems that learn their behaviour in an effective and practical way. In particular, this thesis addresses the first two questions and the others are left as future work.

(i) *How to learn dialogue policies on large state-action spaces.* Most of the available dialogue optimization methods use tabular or function approximation reinforcement learners with a flat setting. The former works well on small/medium size search spaces. The latter has been shown to be feasible on very large ones but with limited convergence guarantees. It remains to be investigated if the tabular approach can be scalable; a potential direction to follow is hierarchical approaches.

(ii) *How to incorporate prior knowledge into optimized dialogue behaviour.* Previously proposed optimization approaches perform learning on constrained and unconstrained search spaces. However, there is a lack of a principled approach for adding constraints to dialogue behaviours before and after learning. This limits the practical application of reinforcement learning dialogue systems in real environments. Thus, effective methods for learning and updating behaviours where required remain to be investigated.

(iii) *How to learn scalable and robust dialogue strategies.* Previous work in the field has been divided into learning dialogue strategies under certainty, and planning under uncertainty. Due to the fact that both research efforts aim to contribute towards adaptive and robust spoken dialogue behaviours, a thorough integration of efforts still remains to be explored.

(iv) *How to learn dialogue strategies for complex behaviours.* Most reinforcement learning dialogue agents so far have optimized confirmation, initiative, and database queries. But other dimensions require further investigation to endow dialogue systems with smarter behaviours; for example: learning to give help, learning to ground, learning to clarify, learning to negotiate, learning to present information, learning to recover from errors. Furthermore, the integration of a wide range of optimized behaviours into a single unified framework also remains to be explored.

(v) *How to simulate conversational environments for dialogue strategy learning.* Although important advances have been made in human-machine dialogue simulation, it is still not very clear how realistic spoken dialogue behaviour can be simulated. Consequently, there is no agreement on how to evaluate the effectiveness of models for simulating dialogue behaviour. The limitations mentioned in the previous section suggest that simulation methods and evaluation metrics need further investigation. Their strengths and weaknesses could be assessed so as to propose more effective and practical alternatives.

(vi) *How to learn dialogue strategies online with real users.* A main limitation of the dialogue strategy optimization approaches proposed so far is that learning is very slow. This issue has motivated researchers to perform learning in an offline fashion: once dialogue strategies have been optimized they are put into operation with frozen optimization. This means that spoken dialogue systems with optimized policies employ static behaviours. An alternative approach is to employ dynamic behaviours – dialogue strategies that can be dynamically improved over time. This would require some sort of lifelong learning approach, where very efficient and effective learning methods would be valuable.

All these research questions aim to contribute to the development of spoken dialogue systems with more sophisticated dialogue behaviours. A learning method solving the problems described above is still a major challenge in this field. As a conclusion, due to the fact that hierarchical reinforcement learning approaches have received very little attention, this thesis will now narrow down its scope to investigate such approaches for spoken dialogue systems.

2.6 Summary

This literature review chapter described previous work in the field of reinforcement learning for spoken dialogue systems. After a brief introduction to reinforcement learning, four approaches for dialogue strategy learning were described based on Markov Decision Processes (MDPs), Partially Observable MDPs (POMDPs), function approximation, and evolutionary reinforcement learning. In addition, this chapter surveyed approaches for simulating the users' dialogue behaviour. It also discussed current practices for evaluating learnt dialogue behaviours and simulated dialogues. Finally, some current research gaps in the field were described. In this literature review it was found that dialogue strategy learning on large search spaces is a critical issue that plays an important role in the development of large-scale spoken dialogue behaviours.

Chapter 3

Hierarchical reinforcement learning: a perspective on spoken dialogue

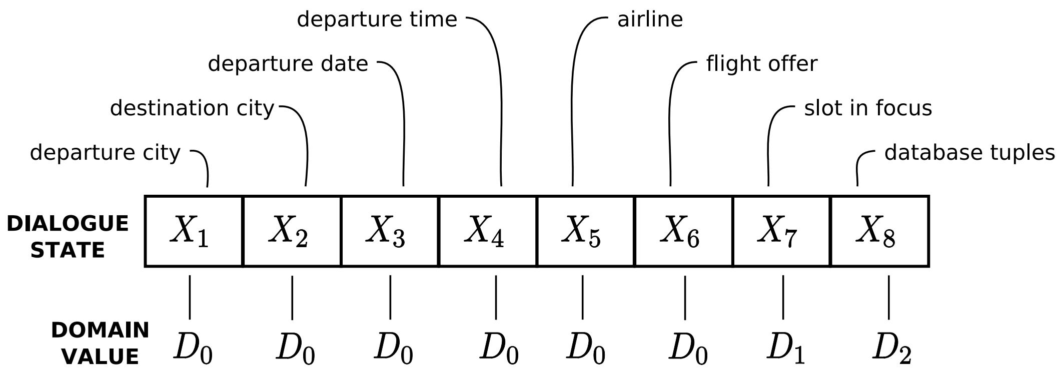
This chapter reviews the literature of hierarchical reinforcement learning – using a number of worked examples – from the perspective of the design of spoken dialogue strategies. Section 3.2 reviews two of the most used methods for hierarchical reinforcement learning, and comments on some recent extensions. Section 3.3 gives an introduction to the Semi-Markov decision process formalism for hierarchical reinforcement learning. Section 3.4 summarizes the current state in the field. Section 3.5 discusses the strengths and weaknesses of such methods for their potential application to large-scale spoken dialogue systems. Finally, the last section summarizes this chapter.

3.1 Introduction

A critical problem in flat reinforcement learning is scalability since it operates with a single policy that behaves by executing only primitive actions. The size of state spaces grows exponentially with the number of state variables incorporated into the environment state – the 'curse of dimensionality'. As a result, reinforcement learning agents find solutions only very slowly. **Temporal abstraction** addresses these problems by incorporating hierarchical structures into reinforcement learning agents. This is attractive for dialogue systems for several reasons. First, human decision-making activity occurs in sequential courses of action, where decisions do not happen at each step, but rather in temporally extended activities following their own policies until termination (Barto and Mahadevan, 2003). Second, hierarchical decision makers can solve more complex problems than flat ones (Dietterich, 2000a). Third, task-oriented dialogues have shown evidence of following hierarchical structures (Grosz and Sidner, 1986; Litman and Allen, 1987; Clark, 1996). This chapter reviews the literature of hierarchical reinforcement learning, including the perspective of dialogue strategy learning.

3.1.1 An illustrative decision-making problem

Consider that you have the task of designing a spoken dialogue strategy for a flight booking system. In such a system the user can say things such as *'a flight from London to Prague for the twenty second of October in the morning travelling with KLM'* – alternatively, the user may provide the information across several shorter utterances. A dialogue strategy is a mapping from dialogue states to actions and specifies the system's way of behaving. The dialogue state (used to describe the current situation in the conversation) is defined by a vector of state variables as illustrated in Figure 3.1. This decision-making problem has 281250 states ($\prod_{X_i \in X} |X_i|$). A sample dialogue using this state space is shown in Table 3.1. The mapping from states to actions can be done either manually, or using flat reinforcement learning methods as described in chapter 2, or using hierarchical reinforcement learning methods as described in the rest of this thesis. The benefits of reinforcement learning include automatic design and optimal behaviour according to a performance measure, while hierarchical methods aim to have a more practical application for systems with large state-action spaces.



$$D_0 = \begin{cases} 0 & \text{if} \quad \text{unknown slot} \\ 1 & \text{if} \quad \text{known with low confidence} \\ 2 & \text{if} \quad \text{known with medium confidence} \\ 3 & \text{if} \quad \text{known with high confidence} \\ 4 & \text{if} \quad \text{confirmed slot} \end{cases} \quad D_1 = \begin{cases} 0 & \text{if} \quad \text{departure city (mandatory)} \\ 1 & \text{if} \quad \text{destination city (mandatory)} \\ 2 & \text{if} \quad \text{departure date (mandatory)} \\ 3 & \text{if} \quad \text{departure time (mandatory)} \\ 4 & \text{if} \quad \text{airline (optional)} \\ 5 & \text{if} \quad \text{flight offer (terminal)} \end{cases} \quad D_2 = \begin{cases} 1 & \text{if} \quad \text{none} \\ 2 & \text{if} \quad \text{few} \\ 3 & \text{if} \quad \text{many} \end{cases}$$

Figure 3.1: *Dialogue state for the flight booking spoken dialogue strategy. Each variable X_i with domain values D_0 has five possible values, variable X_7 has six possible values, and variable X_8 has 3 possible values, resulting in* $5^6 \times 6 \times 3 = 281250$ *states.*

Table 3.1: *Sample human-machine dialogue in the flight booking domain, where the dialogue state is formed by the state variables shown in Fig. 3.1, and a set of actions is available per state. At this point action-selection is arbitrary (before learning).*

Agent	Dialogue State	Action	Sample Utterance
SYS USR	0,0,0,0,0,0,0,1	req	Tell me your flight information. *A flight from Edinburgh to Paris.*
SYS USR	3,2,0,0,0,0,0,1 3,2,0,0,0,0,2,1	acc mic+req	 OK, a flight from Edinburgh to Rome, when do you want to travel? *I said 'to Paris'.*
SYS USR	4,3,0,0,0,0,2,1	sic+req	OK, a flight to Paris, when do you want to fly? *December first, flying with British Airways.*
SYS USR	4,4,1,0,2,0,2,1	apo+req	I'm sorry, when do you want to travel? *On the first of December.*
SYS USR	4,4,3,0,2,0,2,1 4,4,3,0,2,0,3,1	acc mic+req	 Travelling on the first of December with British Airways, at what time do you prefer your flight? *In the morning.*
SYS USR	4,4,4,1,4,0,3,1	apo+req	Sorry, at what time do you prefer to fly? *In the morning.*
SYS USR	4,4,4,3,4,0,3,1	sec	Did you say in the morning? *Yes please.*
SYS USR	4,4,4,4,4,0,5,1	dbq+sta	Please wait while I retrieve information ...
SYS USR	4,4,4,4,4,0,5,2	pre+ofr	All right, flights from Edinburgh to Paris on the first of December...[present information] Which choice do you prefer? *The second one please.*
SYS USR SYS	4,4,4,4,4,2,5,2 4,4,4,4,4,4,5,2	sec	Did you say choice two? *That's right.* [terminal state]

Notes: This example is used for illustration purposes and shows basic behaviour of more complex interactions. Abbreviations: req=request, acc=accept w/high confidence, mic=multiple implicit confirmation, sic=single implicit confirmation, apo=apology, sec=single explicit confirmation, dbq=database query, sta=status of dialogue, pre=present info, ofr=offer choices.

3.1.2 Temporal abstraction for dialogue strategy learning

A learning agent using flat decision-making is limited to primitive actions such as those shown in Table 3.1. In contrast, an agent using temporal abstraction can choose both primitive and composite actions, which are temporally extended actions corresponding to sub-dialogues. For example, the last two dialogue turns can be seen as the sub-dialogue 'presentFlightInfo' that presents information and fill/confirm the terminal slot. In general, hierarchical reinforcement learning agents embrace properties such as abstraction, modularity and reusability that are lacked by flat reinforcement learners.

Abstraction can be defined as the act of removing detail from a concept or object. Abstraction can be divided into temporal abstraction and state abstraction; the latter is addressed in the next subsection. *Temporal abstraction* refers to temporally extended courses of action, where details of complex actions are ignored and treated as composite activities such as the 'presentFlightInfo' action. They can help to explore the search space more quickly. Examples of procedural abstraction include macros, subroutines, abstract actions, composite actions, subpolicies, options, behaviours and subtasks. Such abstractions can be arranged into nested actions, forming a hierarchy of actions at different levels of granularity.

Modularity refers to a divide-and-conquer approach, where a learning problem is decomposed into sub-problems, and the subsolutions are merged into an overall solution. For example, the flight booking dialogue strategy can be decomposed into the sub-behaviours 'getMandatorySlots', 'getOptionalSlots' and 'presentFlightInfo', where they could be subsequently decomposed, and so on. The modular behaviours make internal decisions, independent of external information; Dayan and Hinton (1992) refer to this as 'information and reward hiding'. Modularity is also an important property for enhancing the maintainability and testing of learnt dialogue behaviours.

Reusability occurs when sub-behaviours are shared by multiple parent behaviours. A key idea is that behaviours do not need to learn everything ab initio (i.e. from the beggining); instead, they can be based on previously learnt ones, and possibly reused by other ones. For example, the dialogue sub-behaviour 'presentFlightInfo' could be reused by other spoken dialogue systems. When behaviours are reused in a new problem, the learning speed is accelerated (Dietterich, 2000a).

These properties are crucial for learning the behaviour of large-scale spoken dialogue systems, where the dialogue state may be described using a large set of state variables and/or the dialogue system may have support for a large number of actions.

3.1.3 State abstraction for dialogue strategy learning

The role of state abstraction is to compress the state, assuming that the learning agent does not need to know all the knowledge in every state to take the best actions. The importance of state abstraction is to find solutions on a more compact state representation (also referred to as ‘abstract state space’) that focuses on relevant parts of the state and ignore irrelevant ones, e.g. the dialogue states used to get flight information can ignore the state space for presenting information. State abstraction is a key concept in hierarchical reinforcement learning in order to overcome the problem of the curse of dimensionality. Dietterich (2000a) proposed the following types of state abstraction:

1. *Irrelevant variables*: a state variable is irrelevant for primitive or composite action *a* if it does not affect the cumulative reward of the remaining variables, e.g. the composite actions ‘getMandatorySlots’ and ‘getOptionalSlots’ can ignore the variables of the composite action ‘presentFlightInfo’. In addition, lower-level composite actions have fewer relevant variables than higher-level actions.

2. *Funnel abstractions*: an action is described as a funnel if it causes a large set of states to change into a small set of next states. In this way, the domain value of a state variable is irrelevant for composite action *a* if it does not affect the cumulative reward of the remaining domain values. For example: when the composite action ‘getMandatorySlots’ is executed, the domain values for unfilled and confirmed slots are relevant to the parent, but the remaining values (e.g. the status of such slots, filled with different recognition confidence levels) could be ignored.

3. *Structural constraints*: if a composite action terminates in state *s* or if the termination of a child composite action involves the termination of its parent, then there is no need to store values for that state. For example: the composite action ‘getMandatorySlots’ can ignore values at terminal states.

The first type of abstraction has a wider application in HRL algorithms, and the latter types require the decomposition of the value function to guarantee *safe abstraction*. State abstraction is safe if the learnt policy in the abstract space is also optimal in the original state space. Previous work has applied state abstraction to small-scale problems (Dietterich, 2000a; Andre and Russell, 2002; Uther, 2002; Jong and Stone, 2005; Marthi et al., 2006; Jonsson, 2008), and it remains to be investigated in problems with large sets of state variables. Alternatively, abstractions may be provided by the system developer (Marthi, 2006), but there are no guarantees for safe abstractions.

3.2 Hierarchical reinforcement learning approaches

The incorporation of hierarchical structures into reinforcement learning agents was also motivated by hierarchical planners such as HTN (Hierarchical Task Network), which use a plan library with high-level activities decomposed into lower-level ones (Sacerdoti, 1975; Currie and Tate, 1991). The strenghts of hierarchical reinforcement learning methods in comparison to flat methods are that they find solutions faster and can solve more complex problems. The reinforcement learning methods described in this section are based on the Semi-Markov Decision Process (SMDP) model, a generalisation of the MDP model. An SMDP is a mathematical model for sequential decision-making in temporally extended courses of action, and provides the fundamental theory for hierarchical reinforcement learning agents. SMDPs allow actions to take a variable amount of time to complete, resulting in state transitions with large steps, see Figure 3.2. This allows the agent to explore the search space more efficiently.

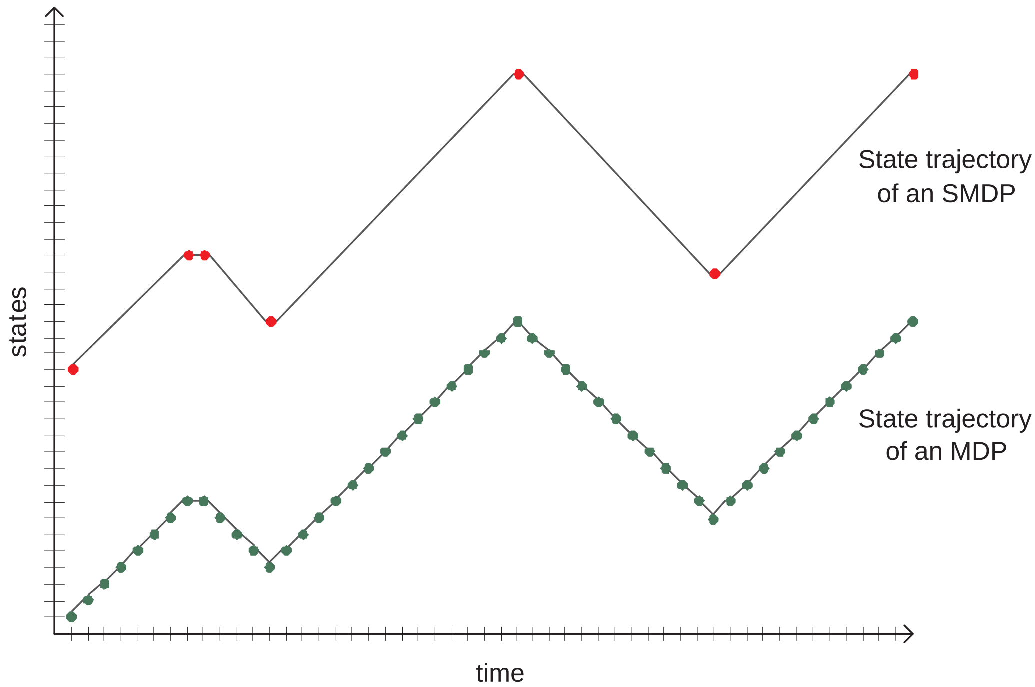


Figure 3.2: *Illustration of state trajectories in MDPs and SMDPs.*

Work on hierarchical reinforcement learning can be broadly classified into agents that learn *context-dependent policies* and those that learn *context-independent policies*. This section reviews approaches for context-dependent policies such as HAMs, and context-independent policies such as MAXQ. It also discusses their strengths and limitations, and briefly comments on recent extensions.

A main weakness of hierarchical reinforcement learning algorithms is that they only produce sub-optimal solutions. The loss in optimality may be due to the following reasons: (1) in a composite action only a subset of primitive actions is allowed; (2) a composite action depends on the execution of child behaviours; (3) unsafe state abstractions; and (4) the prior knowledge included in the policy's behaviour. The SMDP-based hierarchical reinforcement learning algorithms described in the rest of this section may suffer from some of these sub-optimalities, and can be classified according to their type of policy: context-dependent or context-independent[1].

- An optimal **context-dependent policy** achieves the highest cumulative reward among all policies consistent with the given hierarchy. Here, temporally extended behaviours execute actions that may be locally sub-optimal but that are optimal for the other behaviours. The HAMs method learns this kind of policies.

- An optimal **context-independent policy** achieves the highest cumulative reward for the given composite action, but suffers from an additional source of sub-optimality, locally optimal policies. Here, temporally extended behaviours execute actions that are locally optimal but that may be sub-optimal for the other behaviours. The MAXQ method learns this kind of policy.

Dietterich (2000a) points out that there is a trade-off between both types of policy. On the one hand, context-independent policies facilitate state abstraction and policy reuse, but they are only locally optimal. On the other hand, context-dependent policies allow stronger optimality, but they are weaker for state abstraction and policy reuse.

3.2.1 Hierarchical abstract machines

A Hierarchical Abstract Machine (HAM) is a partial program that constrains the actions that a reinforcement learning agent can take in each state (Parr and Russell, 1997; Parr, 1998). HAMs are similar to non-deterministic finite state machines (FSMs) whose transitions may invoke lower-level machines, each machine specifying a sub-dialogue. Figure 3.3 provides a graphical illustration of reinforcement learning with HAMs. In contrast to standard reinforcement learning, here the environment is modelled by an induced SMDP, where the HAM tells the SMDP the available set of actions per state. The learning agent has to optimize decision-making of low and high-level actions taking into account both the environment state s and the machine state $\bar{s}$.

[1]In machine learning jargon, optimal context-dependent policies are known as 'hierarchical optimal policies', and optimal context-independent policies as 'recursively optimal policies'.

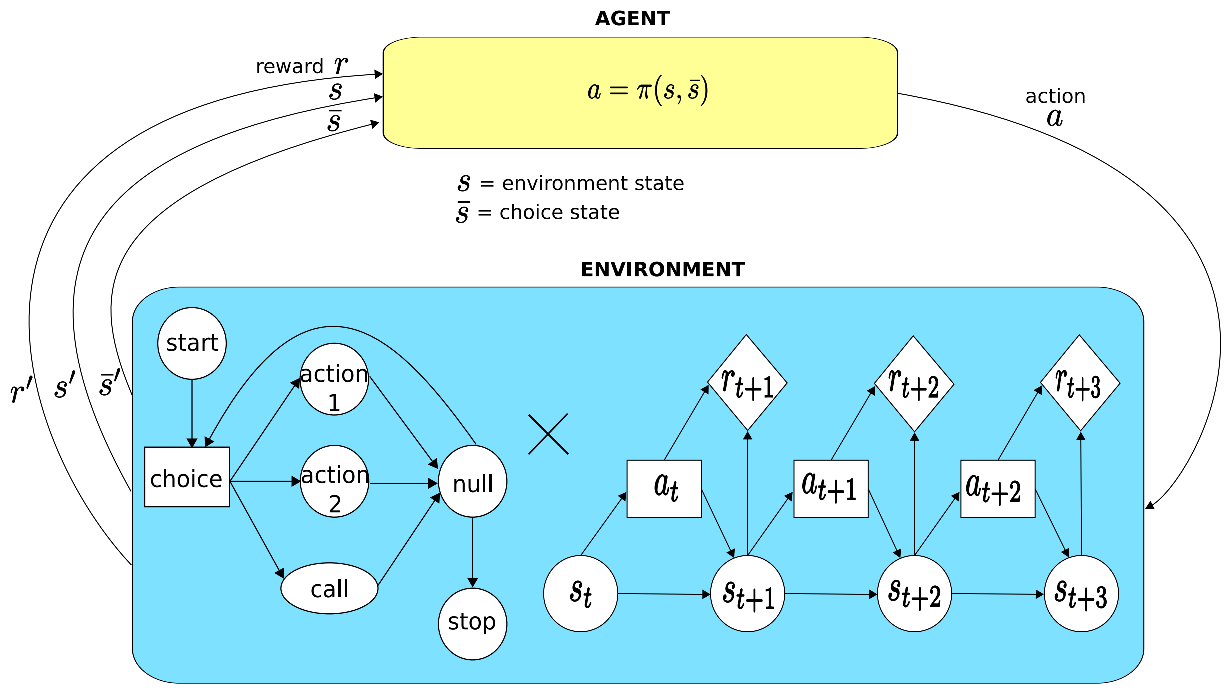


Figure 3.3: *Architecture of the agent-environment interaction for HAM-based reinforcement learning, where a HAM tells the agent the available actions per state.*

A key idea is that the system developer specifies a partial policy and leaves the unspecified part to the reinforcement learning agent. Such prior expert knowledge guides the learning agent through a smaller search space to find solutions much faster than using blind search, because it focuses learning on the parts that were left unspecified. In the flight booking dialogue strategy, one may think of actions that are easy to specify, such as asking for information when slot values are unknown, and also think of actions less easy to specify, such as confirming or reasking for slots previously filled.

A HAM is defined by three elements: (1) a finite set of machine states; (2) an initial state or a start function determining the initial machine state; and (3) a transition function to determine the next state using either deterministic or stochastic transitions. The types of machine state are:

- *start*: execute the current machine (e.g. ‘root’),
- *action*: execute an action (e.g. ‘request departure city),
- *call*: execute another machine (e.g. ‘presentFlightInfo’),
- *choice*: select the next machine state, and
- *stop*: halt execution and return control.

HAMs are appealing for specifying the dialogue behaviour of conversational systems because they can be used for fully-deterministic behaviour, fully-learnt behaviour, or a combination of both. HAMs control the dialogue in a modular way, where each machine in the hierarchy specifies a sub-dialogue. Whilst the root and non-terminal machines execute actions at different levels of granularity (i.e. they may call other machines), the terminal children machines only execute primitive actions (i.e. they do not invoke other machines). The dynamics in a HAM are as follows: when a lower-level machine is called, control is transferred to the start state, where machine states are visited until reaching a stop state, which returns control to the caller, and then determines the next machine state, and so on until reaching the stop state of the root machine.

Figure 3.4 shows a sample HAM that partially specifies the dialogue behaviour for the flight booking system. It uses a root machine that invokes three lower-level machines: 'getMandatorySlots', 'getOptionalSlot' and 'presentFlightInfo'. An assumption in HAM-based controllers is that the stochastic behaviour is not easy to specify and hence requires optimization – so they can be seen as learning values for the stochastic transitions and will prefer the actions with higher value. This HAM focuses on optimizing a confirmation strategy, where stochastic action-selection is used in the root and the children machines. Alternatively, only deterministic action-selection may be used in the parts that do not require to be included in the optimization. The rest of this subsection explains how a HAM-based reinforcement learner optimizes action-selection in choice states $\overline{s_i}$.

For any MDP M and any HAM H, there exists an **induced SMDP** $M' = H \circ M$. The solution defines an optimal policy that maximizes the expected total reward by a reinforcement learning agent executing H in M. The construction of M' is as follows:

(i) The state set is the cross-product of the choice states of H and the environment states of M. Notice that not all pairs of environment-machine state $(s_i, \overline{s_i})$ will be possible, and therefore the learning task becomes easier. An ad hoc algorithm $reduce(H,M)$ can be used to remove inappropriate pairs and states with a single action because they do not require optimization.

(ii) The action set is derived from the stochastic actions in H. Such actions correspond to either action states (primitive actions) or call states (composite actions). The induced state-action space for the flight booking dialogue strategy is shown in Table 3.2, which involves both composite and primitive actions. This table omits deterministic actions because they have only one available action per state.

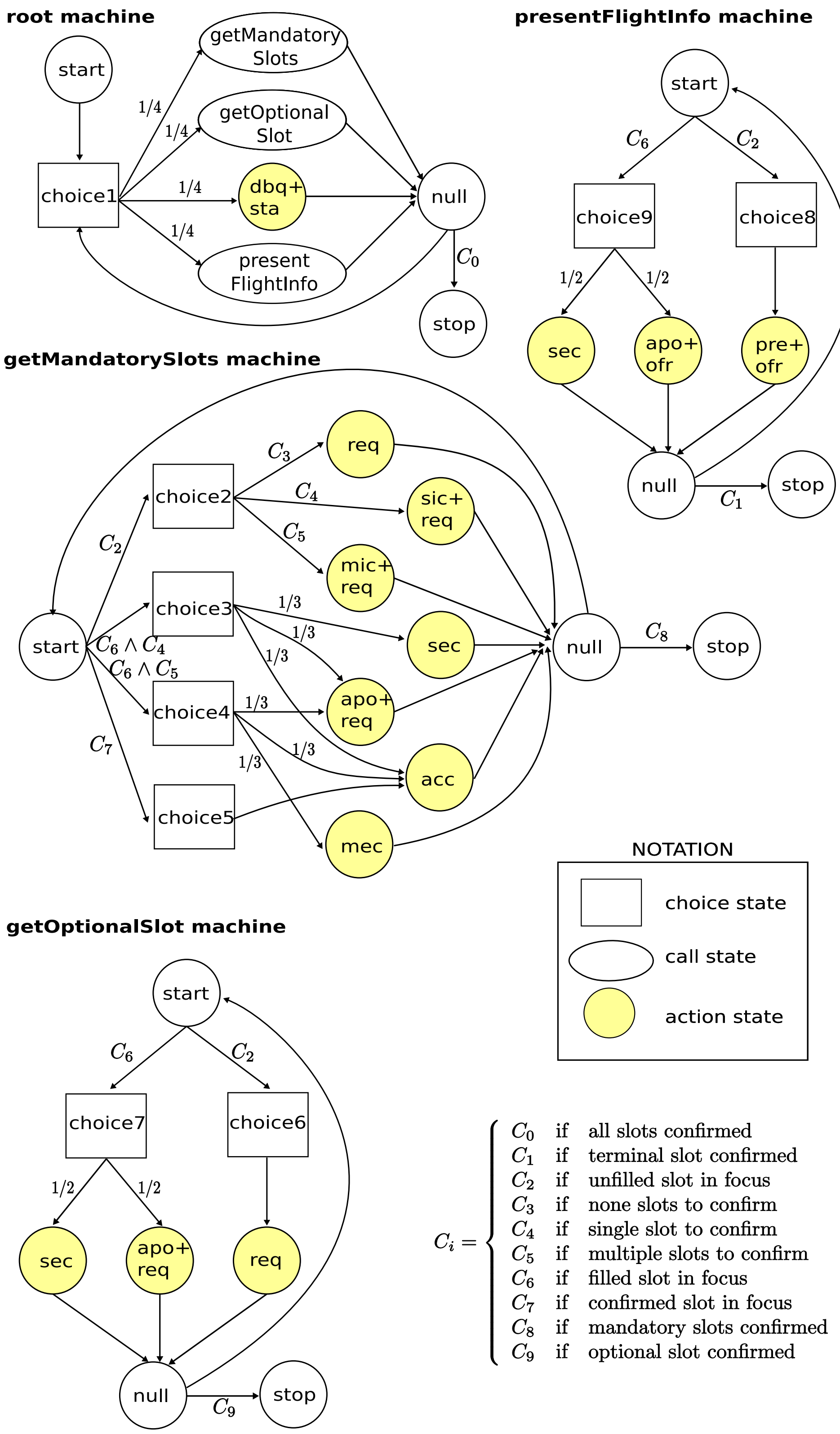


Figure 3.4: *Hierarchical Abstract Machine (HAM) for the flight booking dialogue system. The decision-making points are in machine choice states with deterministic or stochastic choices. Abbreviations: req=request, acc=accept w/high confidence, mic=multiple implicit confirmation, sic=single implicit confirmation, apo=apology, mec=multiple explicit confirmation, sec=single explicit confirmation, dbq=database query, sta=status of dialogue, pre=present information, ofr=offer choices.*

Table 3.2: *Induced state-action space resulting from the cross product of the environment states of Figure 3.1 and choice states of the HAM shown in Figure 3.4. The rest of the state-action pairs are omitted because they have one available action per state.*

Induced State	Induced Action Set
(s, $\bar{s}$=choice1)	getMandatorySlots, getOptionalSlot, presentFlightInfo, dbq+sta
(s, $\bar{s}$=choice3)	sec, apo+req, acc
(s, $\bar{s}$=choice4)	mec, apo+req, acc
(s, $\bar{s}$=choice7)	sec, apo+req
(s, $\bar{s}$=choice9)	sec, apo+ofr

Note: An induced state is also referred to as 'environment-machine state'.

(iii) The state transition function corresponds to executing in parallel the transition function of the MDP and the transition function of the HAM.

(iv) The reward function is defined as $R'([s,\bar{s}],a) = R(s,a)$ if a is an action state, otherwise the reward for non-action states in the HAM is zero. In addition, for each transition from state s_t to state $s_{t+\tau}$ the learning agent computes the cumulative discounted reward $r = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{\tau-t-1} r_{t+\tau}$.

In general, the cross product of HAM and MDP described above results in a Semi-Markov Decision Process (SMDP) because actions take a variable amount of time to complete their execution. A sample dialogue using environment and machine states $(s,\bar{s})$ is shown in Table 3.3, this is equivalent to the one shown in Table 3.1. They differ in the kind of actions they take: whilst a HAM-based dialogue agent takes primitive and composite actions, the former dialogue strategy only takes primitive ones.

The HAMQ-Learning algorithm can be used to learn a HAM-based policy (Parr, 1998). This algorithm uses an extended Q-table $Q([s,\bar{s}],a)$ indexed by an environment state s, machine state $\bar{s}$, and action a, see algorithm 2. A sample call for our case study corresponds to $HSMQ([0,0,0,0,0,0,0,1],root)$. HAMQ-Learning converges under similar conditions as Q-Learning (Watkins and Dayan, 1992). A similar algorithm called 'SMDP Q-Learning' uses only environment states (Bradtke and Duff, 1994), and has been applied to the *options*[2] framework (Sutton et al., 1999; Precup, 2000).

[2]The Options approach changes the notion of actions $a \in A$ in an MDP for options $o \in O$, where an option can last τ time steps (Sutton et al., 1999; Precup, 2000). An option is defined as a 3-tuple $o = <I,\pi,\beta>$, characterized by an initiation set of states $I \subseteq S$, a policy $\pi : S \times A \rightarrow [0,1]$, and a termination condition $\beta : S \rightarrow [0,1]$ that specifies the probability for transitioning to a terminal state.

Table 3.3: *Sample HAM-based dialogue in the flight booking domain using induced dialogue states* $(s, \bar{s})$. *The induced states shown in Table 3.2 have stochastic choices (require optimization) and the remaining ones perform deterministic action-selection.*

Agent	Dialogue State	Action	Sample Utterance
SYS	0,0,0,0,0,0,0,1,choice1	getMandatorySlots	
	0,0,0,0,0,0,0,1,choice2	req	Tell me your flight information.
USR			*A flight from Edinburgh to Paris.*
SYS	3,2,0,0,0,0,0,1,choice4	acc	
SYS	3,2,0,0,0,0,2,1,choice2	mic+req	OK, a flight from Edinburgh to Rome, when do you want to travel?
USR			*I said 'to Paris.'*
SYS	4,3,0,0,0,0,2,1,choice2	sic+req	OK, a flight to Paris, when do you want to fly?
USR			*December first, flying with British Airways.*
SYS	4,4,1,0,2,0,2,1,choice4	apo+req	I'm sorry, when do you want to travel?
USR			*On the first of December.*
SYS	4,4,3,0,2,0,2,1,choice4	acc	
SYS	4,4,3,0,2,0,3,1,choice2	mic+req	Travelling on the first of December with British Airways, at what time do you prefer your flight?
USR			*In the morning.*
SYS	4,4,4,1,4,0,3,1,choice3	apo+req	Sorry, at what time do you prefer to fly?
USR			*In the morning.*
SYS	4,4,4,3,4,0,3,1,choice3	sec	Did you say in the morning?
USR			*Yes please.*
SYS	4,4,4,4,4,0,5,1,choice1	dbq+sta	Please wait while I retrieve information ...
USR			
SYS	4,4,4,4,4,0,5,2,choice1	presentFlightInfo	
	4,4,4,4,4,0,5,2,choice8	pre+ofr	All right, flights from Edinburgh to Paris on the first of December...[present info.] Which choice do you prefer?
USR			*The second one please.*
SYS	4,4,4,4,4,2,5,2,choice9	sec	Did you say choice two?
USR			*That's right.*
SYS	4,4,4,4,4,4,5,2,null		[terminal state]

The HAMs approach does not overcome the curse of dimensionality problem, it is only mitigated by reducing the available actions per state. Its practical application is limited to decision-making problems with few state variables, and it needs to be extended with other ideas for scaling to larger problems. Nonetheless, it has the advantage of merging hand-coded and learnt behaviours into a single framework.

Algorithm 2 The HAMQ-Learning algorithm

function HAMQ(state s, state $\bar{s}$) return totalReward
    $\bar{T} \leftarrow$ state transition function of the machine corresponding to call state $\bar{s}$ (e.g. root)
    $\bar{s} \leftarrow start$, $totalReward \leftarrow 0$, $discount \leftarrow 1$
    while $\bar{s}$ is not a stop state **do**
        if $\bar{s}$ is an action state **then**
            Execute action a (corresponding to $\bar{s}$)
            Observe one-step reward r
        else if $\bar{s}$ is a call state **then**
            $r \leftarrow$ HAMQ(s, a), total reward received whilst action $a \leftarrow \bar{s}$ executed
        else if $\bar{s}$ is a choice state **then**
            Choose action state $\bar{s} \leftarrow \pi(s, \bar{s})$ according to an exploration policy
            continue
        else
            Observe next machine state $\bar{s}'$ from $\bar{T}$ (e.g. a choice, null or stop state)
            $\bar{s} \leftarrow \bar{s}'$
            continue
        end if
        $totalReward \leftarrow totalReward + discount \times r$
        $discount \leftarrow discount \times \gamma$
        Observe resulting environment state s'
        Observe resulting machine state from $\bar{T}$
        $Q([s, \bar{s}], a) \leftarrow (1-\alpha)Q([s, \bar{s}], a) + \alpha \left[r + discount \times \max_{a'}([s', \bar{s}'], a')\right]$
        $s \leftarrow s'$
        $\bar{s} \leftarrow \bar{s}'$
    end while
end function

Recent advances in HAM-based hierarchical reinforcement learning are as follows: Andre and Russell (2000) extended HAMs to support parameterized subroutines, temporary interrupts, aborts, and memory variables. Andre and Russell (2002) apply *safe* state abstraction to partial hierarchical programs, written in a language called ALisp. Finally, Marthi et al. (2005) investigated agents that control several effectors simultaneously, and suggested multithreaded partial programs for such a purpose.

3.2.2 MAXQ

In the MAXQ method the system developer specifies a hierarchy of subtasks and the reinforcement learning agent specifies their behaviour. Two versions can be identified in this method. The first decomposes a given Markov Decision Process (MDP) into a hierarchy of Semi-Markov Decision Processes (SMDPs). The second version extends the first by decomposing the value function recursively and by ignoring parts of the state space, resulting in faster learning. In addition, the latter version includes approaches for tackling the sub-optimalities caused by the imposed hierarchy.

3.2.2.1 Hierarchical problem decomposition

In contrast to the HAM-based reinforcement learning method that learns a single policy, the MAXQ method learns multiple policies. In MAXQ, a given Markov Decision Process (MDP) is decomposed into a hierarchy of sub-problems (also referred to as **hierarchy of subtasks**) provided by a system developer (Dietterich, 2000b). In the context of spoken dialogue systems a subtask corresponds to a sub-dialogue, i.e. each sub-dialogue is controlled by a separate policy. A sample hierarchy of sub-dialogues for the flight booking dialogue strategy is shown in Figure 3.5. When a parent subtask invokes a child subtask, control is transferred to the child; when it terminates its execution, control is returned to the parent subtask. In this method solving the hierarchical decision making problem means finding an optimal policy for the root subtask.

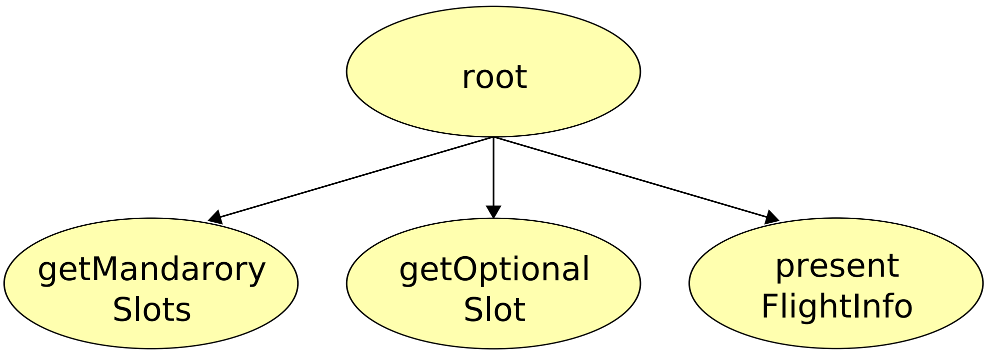


Figure 3.5: *Top-down hierarchy of subtasks for the flight booking system. A parent subtask can invoke child subtasks, when they terminate, control is returned to the caller.*

The hierarchical decomposition allows to find **context-independent policies**, and has the following advantages: (1) policies learnt in child subtasks can be reused by parent subtasks, (2) value functions learnt in subtasks can be shared so that learning in other subtasks is accelerated, and (3) value functions can be represented in a more compressed way by applying state-action abstraction, which ignores irrelevant parts of their corresponding state-action space.

In this method a given MDP M is decomposed into a set of subtasks $\{M_0, M_1, ..., M_n\}$. Each subtask M_i defines a Semi-Markov Decision Process (SMDP) characterized by a set of states, a set of actions, a state transition function, and reward function. The action set includes either primitive actions lasting a single time step, or composite actions corresponding to subtasks that last for multiple time steps. For example, Table 3.4 shows the actions available per dialogue subtask in the hierarchy of subtasks shown in Figure 3.5. Note that each dialogue subtask uses its own set of actions.

Table 3.4: *Action spaces in the hierarchy of dialogue subtasks for the flight booking system, where the root subtask uses both composite and primitive actions and the child subtasks use only primitive actions. See Table 3.1 for a description of primitive actions.*

SubtaskID	Subtask	Actions available
M_0	root	getMandatorySlots, getOptionalSlot, presentFlightInfo, dbq+sta
M_1	getMandatorySlots	req, apo+req, sic+req, mic+req, sec, mec, acc
M_2	getOptionalSlot	req, apo+req, sec
M_3	presentFlightInfo	pre+ofr, apo+ofr, sec

A sample hierarchical dialogue using the subtasks above is shown in Table 3.5. Although the dialogue states use the same information in each subtask, they can ignore some parts of the state space in order to find the solution on a more compact search space than the original one. This property is referred to as ‘state abstraction’ (see section 3.1.3). The solution for the hierarchy of SMDPs is a hierarchical policy containing one locally optimal policy per subtask in the problem $\pi^* = \{\pi^*_0, ..., \pi^*_n\}$.

To learn such a hierarchy of policies, Dietterich (2000b) proposed the Hierarchical Semi-Markov Q-Learning algorithm (also referred to as HSMQ-Learning), where the optimal policy performs action-selection according to $\pi^*(i,s) = \arg\max_a Q^*(i,s,a)$. This algorithm updates Q-values as shown in algorithm 3 – line 14, where value $Q(i,s,a)$ denotes the cumulative reward for executing action a in state s of subtask

Table 3.5: *Sample hierarchical dialogue in the flight booking domain. Figure 3.1 describes the dialogue state, and Table 3.4 shows the actions available per subtask.*

Agent	Subtask	Dialogue State	Action	Sample Utterance
SYS	M_0	0,0,0,0,0,0,0,1	getMandatorySlots	
SYS	M_1	0,0,0,0,0,0,0,1	req	Tell me your flight information.
USR				*A flight from Edinburgh to Paris.*
SYS	M_1	3,2,0,0,0,0,0,1	acc	
SYS	M_1	3,2,0,0,0,0,2,1	mic+req	OK, a flight from Edinburgh to Rome,
				when do you want to travel?
USR				*I said 'to Paris.'*
SYS	M_1	4,3,0,0,0,0,2,1	sic+req	OK, a flight to Paris,
				when do you want to fly?
USR				*December 1st, flying with British Airways*
SYS	M_1	4,4,1,0,2,0,2,1	apo+req	I'm sorry, when do you want to travel?
USR				*On the first of December.*
SYS	M_1	4,4,3,0,2,0,2,1	acc	
SYS		4,4,3,0,2,0,3,1	mic+req	Travelling on the first of December with
				British Airways, at what time do you
				prefer your flight?
USR				*In the morning.*
SYS	M_1	4,4,4,1,4,0,3,1	apo+req	Sorry, at what time do you prefer to fly?
USR				*In the morning.*
SYS	M_1	4,4,4,3,4,0,3,1	sec	Did you say in the morning?
USR				*Yes please.*
SYS	M_1	4,4,4,4,4,0,5,1		[terminal state of subtask M_1]
SYS	M_0	4,4,4,4,4,0,5,1	dbq+sta	Please wait while I retrieve information...
USR				
SYS	M_3	4,4,4,4,4,0,5,2	presentFlightInfo	
	M_3	4,4,4,4,4,0,5,2	pre+ofr	All right, flights from Edinburgh to Paris
				on the first of...[present information]
				Which choice do you prefer?
USR				*The second one please.*
SYS	M_3	4,4,4,4,4,2,5,2	sec	Did you say choice two?
USR				*That's right.*
SYS	M_3	4,4,4,4,4,4,5,2		[terminal state of subtask M_3]
SYS	M_0	4,4,4,4,4,4,5,2		[terminal state of root subtask]

i (executed using a stack mechanism). The execution of action a lasting τ time steps receives the cumulative discounted reward $r = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^{\tau-t-1} r_{t+\tau}$. The dynamics of subtasks are as follows: when a subtask terminates, it is popped off the stack, and control is transferred to the next available subtask in the stack, and so on until popping off the root subtask. A subtask terminates when it reaches one of its terminal states. This algorithm is executed until the Q-values of the root subtask stabilize.

Algorithm 3 The HSMQ-Learning algorithm

function HSMQ(state s, subtask i) **return** $totalReward$
    $totalReward \leftarrow 0$, $discount \leftarrow 1$
    while subtask i is not terminated **do**
        Choose action a from s derived from $Q(i,s)$ (e.g. ε-greedy)
        Execute action a
        if a is primitive **then**
            Observe one-step reward r
        else if a is composite **then**
            $r \leftarrow$ HSMQ(s,a), which invokes subtask a
                and returns the total reward received while a executed
        end if
        $totalReward \leftarrow totalReward + discount \times r$
        $discount \leftarrow discount \times \gamma$
        Observe resulting state s'
        $Q(i,s,a) \leftarrow (1-\alpha)Q(i,s,a) + \alpha\left[r + discount \times \max_{a'} Q(i,s',a')\right]$
        $s \leftarrow s'$
    end while
end function

The HSMQ-Learning algorithm converges to **optimal context-independent policies** if the learning rate parameter α is decayed according to equation 2.10, and if the exploration policies satisfy the following properties: (1) each action is executed infinitely often in every state that is visited infinitely often, and (2) in the limit, the policy is greedy with respect to the Q-value function. This is due to the fact that parent subtasks rely on the behaviour of their children to learn their own optimal behaviour.

3.2.2.2 Decomposition of the value function

Decomposing the value function means splitting the value of states or state-action pairs into multiple values. A key benefit of this decomposition is that it allows parts of the state space to be ignored, and a more compact representation of learnt values to be stored.

The MAXQ value function decomposition splits value functions in a recursive way into two additive values: (1) the value for executing a child action, which may be a primitive action or subtask; and (2) the value after executing such child action until its parent terminates (Dietterich, 2000a). The former are referred to as *projected value functions* $V(i,s)$ and specify the cumulative reward of executing subtask i in state s. The latter are referred to as *completion functions* $C(i,s,a)$ and specify the cumulative reward after executing action a in state s until completing subtask i. A sample MAXQ decomposition for the value function in the flight booking domain is illustrated in Figure 3.6. Note that the value of $V(getMandatorySlots,s)$ can be computed by adding its projected and completion value functions and hence does not require to be stored. For example, a value for the root subtask in state s is computed as

$$V(root,s) = V(req,s) + C(getMandatorySlots,s,req) + C(root,s,getMandatorySlots). \tag{3.1}$$

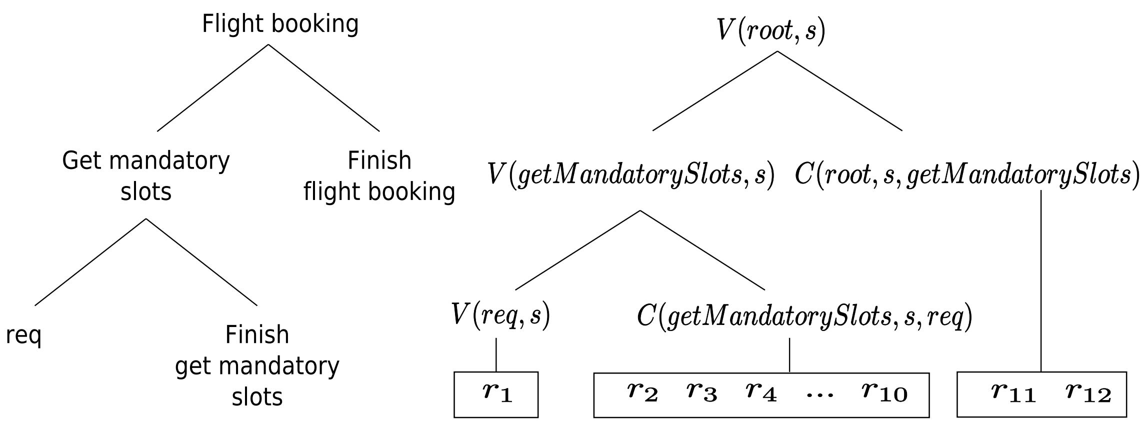


Figure 3.6: *Example of MAXQ value function decomposition for the flight booking dialogue strategy, where the values of state-action pairs are decomposed hierarchically into two values. The left tree uses natural language and the right one uses formal notation. The sequence of rewards r_i is given for executing primitive actions.*

In general, the MAXQ value function decomposition has the form

$$V^{\pi}(a_0,s) = V^{\pi}(a_m,s) + C^{\pi}(a_{m-1},s,a_m) + ... + C^{\pi}(a_1,s,a_2) + C^{\pi}(a_0,s,a_1), \tag{3.2}$$

Algorithm 4 The MAXQ-0 Learning algorithm

```
1: function MAXQ-0(MaxNode i, State s)
2:     if i is a primitive action then
3:         Execute action i, receive r, and observe next state s'
4:         V(i,s) = (1 − α(i))V(i,s) + α(i)r
5:         return 1
6:     else // i is a subtask
7:         let count = 0
8:         while subtask i is not terminated do
9:             Choose action a according to exploration policy π_x(i,s)
10:            let N=MAXQ-0(a,s) (recursive call)
11:            observe next state s'
12:            C(i,s,a) = (1 − α)C(i,s,a) + α(i) · γ^N V(i,s')
13:            count = count + N
14:            s = s'
15:        end while
16:        return count
17:    end if
18: end function
19: //Main program
20: initialize V(i,s) and C(i,s,j) arbitrarily
21: MAXQ-0(root node 0, starting state s_0)
```

where $a_0, a_1, \ldots, a_m$ is the path of subtasks chosen by the hierarchical policy π going from the root subtask a_0 to the primitive action a_m.

The MAXQ-0 algorithm can be used to learn locally optimal policies based on this value function decomposition (see algorithm 4), where the value $V(i,s')$ in line 12 is computed according to

$$V(i,s) = \begin{cases} \max_a(V(a,s) + C(i,s,a)) & \text{if} \quad i \text{ is composite} \\ V(i,s) & \text{if} \quad i \text{ is primitive.} \end{cases} \tag{3.3}$$

A similar algorithm called MAXQ-Q allows the use of arbitrary *pseudo-rewards*, which are used to specify how desirable each terminal state is for the given subtask. A pseudo-reward function – specified by the system developer – typically assigns pseudo-rewards of 0 to non-terminal states and goal terminal states, and negative pseudo-rewards to non-goal terminal states, see (Dietterich, 2000a) for more details.

In summary, the MAXQ method divides an MDP into a hierarchy of SMDPs. It learns a hierarchy of context-independent policies using two types of value functions: *non-decomposed* using the HSMQ-learning algorithm, and *decomposed* using the MAXQ-0 or MAXQ-Q algorithms. The policies allow safe state abstraction, but decomposed-based ones use more compact representations. In addition, this method executes policies in a hierarchical or non-hierarchical way, the latter mitigates the sub-optimalities derived from the imposed hierarchy. The non-hierarchical form of execution requires extra learning by using a mechanism similar to policy iteration (Dietterich, 2000a).

The MAXQ method has been extended to multi-agent and continuous-time hierarchical reinforcement learning, and shown to be feasible in complex scheduling tasks (Makar and Mahadevan, 2001; Ghavamzadeh and Mahadevan, 2001). In addition, Hengst (2003) proposed a bottom-up approach for discovering hierarchies of subtasks by incrementally finding subspace regions collapsed into abstract states, where each subregion corresponds to a different SMDP.

3.3 Semi-Markov Decision Processes

The dynamics of hierarchical reinforcement learning methods can be represented with the Semi-Markov Decision Process (SMDP) formalism, which allows us to model temporally extended actions. The SMDP model was originally formulated as a 5-tuple $<S,A,T,R,F>$ characterized as follows: S is a finite set of states in the environment, A is a finite set of actions, $T(s,a,s')$ is a transition function to the next state s' given the current state s and action a with probability $P(s'|s,a)$, $R(s'|a,s)$ is the reward function that specifies the reward given to the agent for choosing action a when the environment makes a transition from s to s', and $F(\tau|s,a)$ is a function giving the transition duration probability that action a in state s will terminate in τ time units. In SMDPs the duration can take either real or integer values, which correspond to continuous-time SMDPs and discrete-time SMDPs (Putterman, 1994; Mahadevan et al., 1997; Parr, 1998).

Dietterich (2000a) reduced the formulation above for the discrete-time SMDP model to a 4-tuple $<S,A,T,R>$, which extends the transition and reward functions with the random variable τ representing the number of time steps it takes to execute an action a in state s. Following Dietterich's formulation of an SMDP, it needs only to consider actions with integer-value durations. In this way, the transition and reward functions are extended as $P(s',\tau|s,a)$ and $R(s',\tau|s,a)$, respectively. The latter specifies the cumulative discounted reward while executing a. In spoken dialogue, the state transitions can be seen as executing a sub-dialogue a_t (compounded by a sequence of lower-level actions) starting in $s = s_t$ and completing its execution in state $s' = s_{t+\tau}$ (see Figure 3.7).

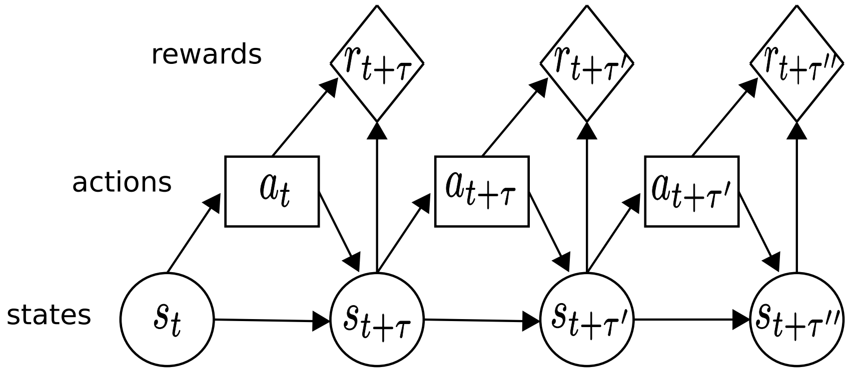


Figure 3.7: *Dynamics in a Semi-Markov decision process, where state transitions and rewards depend on the amount of time taken by actions to complete their execution.*

The Bellman equations for V^* and Q^* for a discrete-time SMDP are rewritten as

$$V^*(s) = \max_a \left[\sum_{s',\tau} P(s',\tau|s,a)[R(s',\tau|s,a) + \gamma^\tau V^*(s')] \right], \tag{3.4}$$

and

$$Q^*(s,a) = \sum_{s',\tau} P(s',\tau|s,a)[R(s',\tau|s,a) + \gamma^\tau \max_{a'} Q^*(s',a')]. \tag{3.5}$$

The solution to a Semi-Markov decision process is a policy π mapping states to actions. An optimal policy is defined similarly to the MDP model, where π is optimal if and only if $\forall_s \pi(s) \in \arg\max_a Q^\pi(s,a)$. It can be learnt either by dynamic programming algorithms applied to SMDPs (Howard, 1971; Putterman, 1994) or by hierarchical reinforcement learning methods such as those described in this thesis.

3.4 Current state of hierarchical reinforcement learning

Briefly, the current state of Hierarchical Reinforcement Learning (HRL) is as follows:

- *Partial observability*: Most HRL approaches assume fully observable states. However, it is well known that many reinforcement learning agents have to model hidden states due to their noisy perceptions of the world. In this context only a few investigations have been reported (Theocarous, 2002; Hansen and Zhou, 2003; Theocarous et al., 2004; Pineau, 2004; Foka and Trahanias, 2007), meaning that the fully observable setting has been developed more extensively.

- *Hierarchy discovery*: Some investigations have attempted to build a hierarchy of temporally extended courses of action (McGovern, 2002; Stolle and Precup, 2002; Hengst, 2003), but they have been applied only to small-scale systems.

- *Dynamic abstraction:* Most HRL approaches use fixed state abstractions. It is known that the relevant state variables for a given behaviour depend on the activity being executed (Jonsson and Barto, 2000). Therefore, dynamic abstraction in HRL approaches would be very valuable, and awaits exploration.

- *Knowledge representation*: Most previously proposed HRL methods employ a vector of numeric state variables to represent the current situation of the world, and also employ actions without complex descriptions. Whilst this form of knowledge representation may be sufficient for slot filling spoken dialogue systems, other forms of knowledge representation may be required for more complex human-machine interactions such as negotiation or collaborative dialogues. For this reason some machine learning researchers have turned their attention to the emerging field known as 'Relational Reinforcement learning' (RRL) that combines reinforcement learning with inductive logic programming (Dzeroski et al., 2001; Tadepalli et al., 2004); a related hybrid hierarchical approach was proposed by (Ryan, 2002). But hierarchical RRL approaches are less mature than those with simpler representations.

- *Large-scale applications*: In tabular HRL, Makar and Mahadevan (2001) employed the MAXQ approach to a state space consisting of 2^{30} states, applied to automated guided vehicles scheduling. The options approach with tile-coding function approximation has been applied to very large state spaces in the RoboCup Soccer domain (Stone et al., 2005). In addition, the HAMs approach with state abstraction and linear function approximation has been applied to the Stratagus computer game consisting of 2^{500} states and 2^{200} actions (Marthi, 2006). This suggests that tabular HRL can be applied to medium or large search spaces, but that HRL with function approximation is the way to address very large ones.

3.5 Discussion

Potentially, any of the currently available hierarchical reinforcement learning methods can be applied to spoken dialogue systems. However, some approaches may be better suited than others for their development, optimization and maintenance. To identify their strengths and weaknesses the following issues are addressed: (1) learning under uncertainty, (2) learning on large search spaces, (3) learning with prior expert knowledge, (4) efficient learning, (5) state abstraction, and (6) optimality.

Firstly, an important requirement of spoken dialogue systems is to learn adaptive behaviour under noisy perceptions such as the speech recognition and understanding modules. However, hierarchical methods that learn with partially-observable states are yet not as developed as those that learn with fully-observable ones. Furthermore, Mahadevan et al. (2004) points out that there is less uncertainty at higher levels of the hierarchy. An example in context is as follows: a dialogue system is more sure of which dialogue goal it is in, rather than exactly what has been said in the current user utterance. In addition, tabular HRL under the SMDP model has not been applied before to spoken dialogue. As was noted in the previous section, it can be applied to problems with medium or reasonably large search spaces. These arguments make it worth investigating the SMDP model, which learns hierarchical dialogue behaviours under certainty.

Secondly, whilst any of the SMDP-based approaches can support very large search spaces using function approximation, the MAXQ framework is the only tabular approach that can overcome the curse of dimensionality problem. This is possible by decomposing the target MDP into a hierarchy of SMDPs and by applying state abstraction in each subtask. Therefore, from all tabular approaches the MAXQ framework is the most appealing for learning spoken dialogue strategies with large state-action spaces. The hierarchical decomposition may not only yield faster optimizations, but perhaps also facilitates their maintenance and reusability.

Thirdly, another potentially relevant requirement for specifying the behaviour of spoken dialogue systems is to allow some hand-crafted behaviour rather than purely learnt. From the approaches above, HAMs is the only approach that provides a principled framework to incorporate prior expert knowledge into reinforcement learning systems. This is particularly important in order to combine hand-coded dialogue behaviours with optimized ones.

Fourthly, an additional requirement of spoken dialogue systems is to support very fast learning methods in order to learn behaviours from a small set of human-machine conversations. All approaches above require a large number of interactions to find optimal policies. Nevertheless, the HAMs approach provides the fastest learning framework because it applies learning only where necessary, at choice states in the HAM, supplying the SMDP with a reduced action set per state.

Fifthly, the state abstraction methods investigated so far have been applied to decision-making problems with few state variables. Currently, it is not clear how to perform automatic safe state abstraction for dialogue states represented with several tens of state variables resulting in (very) large state spaces. As a first step such state abstractions can be provided by the system developer and tested experimentally in comparison with a baseline system. However, future research should take this issue into account in order to reduce potential sub-optimalities derived from manual abstractions.

Lastly, whilst the HAMs approach aims for hierarchical optimality, the MAXQ approach aims for locally optimal solutions. Although the latter is a weaker form of optimality, that is the price for overcoming the curse of dimensionality. A main assumption in this thesis is as follows: *if the solution is near-optimal and generates dialogues that make sense to humans, then such loss in optimality may be affordable*.

Based on these strengths and weaknesses, the scope of this thesis is narrowed down to hierarchical reinforcement learning under certainty based on the SMDP model. Though the current state-of-the-art in HRL presents significant advances, the scope of this research is narrowed down further to investigate tabular hierarchical approaches to spoken dialogue. From the available HRL approaches, it is hypothesized that HAMs and MAXQ have a high potential application for optimizing large-scale spoken dialogue strategies. This hypothesis is tested experimentally in the chapters that follow.

3.6 Summary

This literature review chapter presented an introduction to fully-observable hierarchical reinforcement learning approaches, and used a number of worked examples for such a purpose. It focused on the Semi-Markov Decision Process (SMDP) model, used for sequential decision-making at temporally extended courses of action, which surprisingly has not been applied to spoken dialogue before. It described two of the most influential hierarchical reinforcement learning methods in the field: HAMs and MAXQ, including recent advances. In addition, this chapter highlighted the current state-of-the-art of hierarchical reinforcement learning. Finally, the HAMs and MAXQ methods were identified as promising for optimizing spoken dialogue strategies for larger-scale systems than those so far attempted.

Chapter 4

A heuristic simulation environment for learning dialogue strategies

This chapter describes a dialogue simulation environment that does not require training data and can be used by reinforcement learning agents to optimize or test spoken dialogue strategies. Section 4.2 overviews the simulation environment at the dialogue act level of communication. Section 4.3 describes each component in the simulation environment (user behaviour, speech recognizer, database) and includes a baseline of machine behaviour. Section 4.4 describes two experimental dialogue systems in the flight booking and travel planning domains. Section 4.5 proposes metrics for evaluating user and machine behaviour, and compares to current literature on user simulations (Schatzmann et al., 2005a). Section 4.6 discusses the strengths and weaknesses of the proposed environment. Finally, section 4.7 gives a summary and draws conclusions.

4.1 Introduction

Human-machine dialogue simulation consists of artificial conversations generated between a spoken dialogue manager and a simulated conversational environment (which includes automated speech and language processing modules, and a simulated user). If a dialogue manager follows a given dialogue strategy, then different strategies can be tested in the simulated conversational environment in order to find better ones (Eckert et al., 1997), and the dialogue strategy can be optimized automatically (Levin and Pieraccini, 1997). For such purposes, the speech and word levels can be ignored: it can be assumed that conversations based on dialogue acts – incorporating a noise model – are sufficient to enable optimal dialogue strategies to be learnt (Young, 2000).

Previous investigations in dialogue strategy optimization use two types of learning environments: corpus-based and simulation-based. The former might be preferred because they display the actual dynamics of real conversations (Walker, 2000; Litman et al., 2000; Roy et al., 2000; Singh et al., 2002). However, their application has been limited to small-scale systems due to the fact that a large number of dialogues is required to find an optimal dialogue strategy. In contrast, simulation-based environments are more practical for generating a large amount of different dialogues without the need for real users (Eckert et al., 1997; Levin et al., 2000; Scheffler and Young, 2000; Lin and Lee, 2001; Pietquin, 2004; Chung, 2004; Filisko and Seneff, 2005; Cuayáhuitl et al., 2005; Georgila et al., 2005a, 2006; Pietquin, 2006; Rieser and Lemon, 2006a; Cuayáhuitl et al., 2006b; Hurtado et al., 2007; Toney, 2007; Schatzmann et al., 2005b, 2007a,c; López-Cózar et al., 2008). The drawback of simulation-based approaches is that they may not generate realistic dialogues. Nevertheless, they can help to find errors in dialogue strategies, and initially to optimize the dialogue strategy.

Currently available simulated user models are mostly data-driven approaches, which need a significant amount of annotated training dialogue data (Eckert et al., 1997; Scheffler and Young, 2000; Pietquin, 2004; Georgila et al., 2005a, 2006; Cuayáhuitl et al., 2005; Hurtado et al., 2007; Schatzmann et al., 2007a,c). Although statistical models are very appealing for dialogue simulation, they have a number of drawbacks including the requirement for costly annotated dialogue data, the difficulty of acquiring sufficient training data, and the fact that the resulting smoothed probability distributions may yield incoherent user behaviour (generated by choosing stochastically from the whole set of user dialogue acts in each dialogue state). Furthermore, there is a lack of agreement about the evaluation of user simulations. Previously proposed evaluation metrics are based on properties for measuring the statistical similarity between simulated and real user behaviour (Schatzmann et al., 2005a; Georgila et al., 2005a, 2006; Cuayáhuitl et al., 2005; Rieser and Lemon, 2006a), or for evaluating user models viewed as predictors of system performance (Williams, 2007a). The following facts suggest that richer metrics for evaluating user simulations are needed: (a) current metrics only report a rough indication of dialogue realism, and (b) they cannot distinguish if a given sequence of machine-user dialogue acts is realistic or not.

This chapter presents a simulation framework for generating and evaluating human-machine conversations based on a heuristic approach. The proposed environment generates coherent and distorted conversations, useful for testing and/or learning dialogue strategies for mixed-initiative multi-goal spoken dialogue systems.

4.2 A heuristic dialogue simulation environment

A simulation environment of human-machine conversations involves modelling the dynamics of everything that is outside the dialogue manager. This chapter proposes an approach for information-seeking dialogue systems that does not require data for training the models in the simulation environment. This approach uses heuristics to simulate the dynamics of task-oriented conversations based on dialogue acts, and uses three simulation models which are shown in the bottom of Figure 4.1. The first simulation model (on the right of the figure) generates coherent user responses, i.e. responses that make sense to humans. Here it was assumed that real users behave in a coherent fashion, based on user dialogue acts that are consistent according to a user Knowledge Base (KB) that keeps the history of the conversation. This is a strong assumption and its validity is addressed later. The second model distorts coherent user dialogue acts due to imperfect speech recognition and understanding. Finally, the third model mimics the database queries and results. The distorted user responses and database results update the machine's KB so that the dialogue strategy can choose actions accordingly.

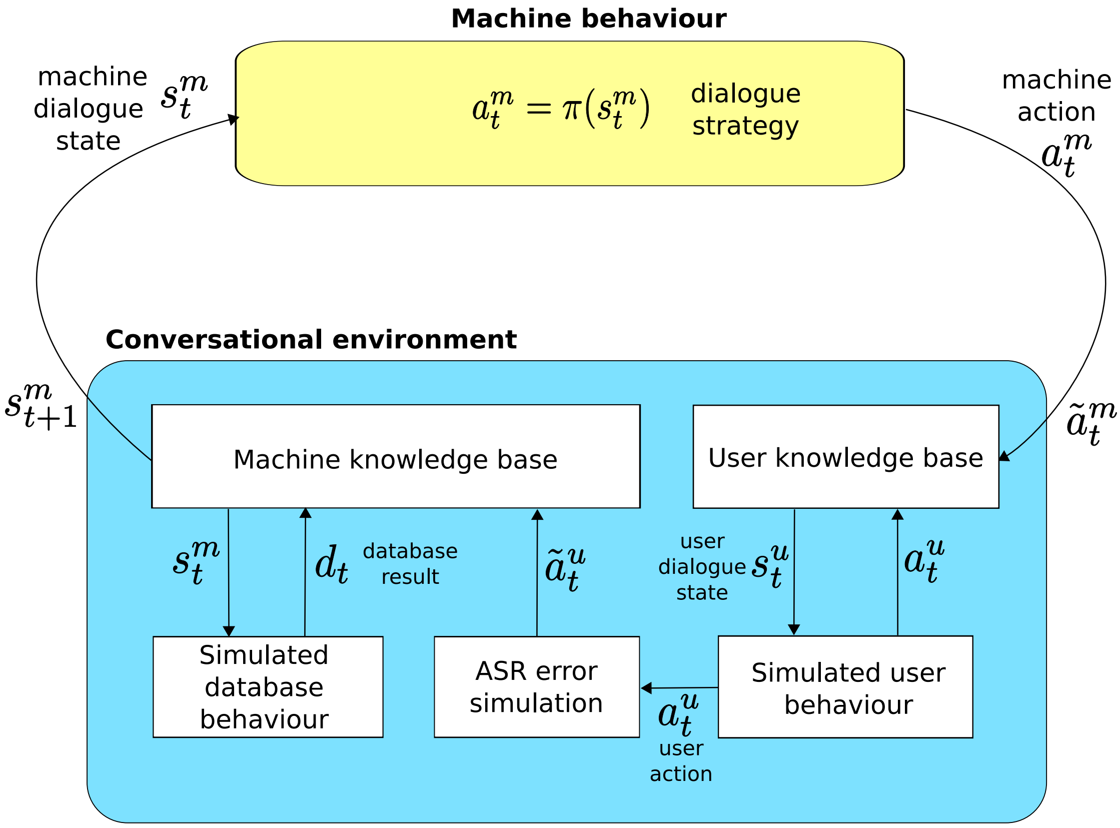


Figure 4.1: *The agent-environment interaction for simulating human-machine conversations, useful for learning or testing dialogue strategies for spoken dialogue systems.*

Figure 4.1 shows the agent-environment interaction for human-machine dialogue simulation (detailed in the next section). The interaction is as follows: the machine is in a given dialogue state s_t^m, and takes dialogue act a_t^m by following dialogue strategy $\pi(s_t^m)$. A distorted machine dialogue act $\tilde{a}_t^m$ (machine response[1]) is fed into the user's KB to observe the user dialogue state s_t^u, from which an action a_t^u is taken (user response). This user response is distorted with ASR errors into $\tilde{a}_t^u$, and is fed into the machine's KB. The machine action may require interaction with simulated database behaviour by sending queries and retrieving database results d_t. Then the next machine state s_{t+1}^m is observed from the machine's current KB. Once the machine is in a new state, it takes another dialogue act, and so on until the end of the conversation.

4.3 Human-machine dialogue modelling

A human-machine dialogue can be modelled by the perceptions and actions of both conversants. Figure 4.2 shows the dynamics of communication at the dialogue act level. The conversants use two sources of knowledge at different levels of granularity: **knowledge-rich states** k_t (also referred to as "knowledge base") to represent all possible perceptions about the conversation, and **knowledge-compact states** s_t to represent a compact version of the current dialogue state. The latter are used for action selection.

The basic elements in a conversation are given by *speech acts*, which represent intentions conveyed to the partner conversant (Austin, 1962; Searle, 1969). This research refers to them as *dialogue acts*, and decomposes them into **dialogue act types** and **dialogue acts**. The former represent types of intentions, e.g. "$a_t^m = confirm$". The latter extend dialogue act types by taking context into account from conveyed slot-value pairs, e.g. "$a_t^m = confirm(date = 01dec2007, time = morning)$".

For task-oriented conversations a small set of dialogue act types can be employed. Table 4.1 shows the core dialogue act types that define the behaviour of human-machine simulated dialogues. The user dialogue act types are a subset of the ones proposed by Georgila et al. (2005b), and the set of machine dialogue act types are an extension of the ones employed by Walker and Passonneau (2001). In addition, a single utterance may convey composite dialogue acts, which are compounded by multiple dialogue act types with their corresponding slot-value pairs. Because the number of unique dialogue acts is usually large, in the proposed simulation framework both conversants choose actions based on dialogue act types, given in context to the partner conversant.

[1] The reason for distorting machine responses (fed to the user's KB) was to model user confusions.

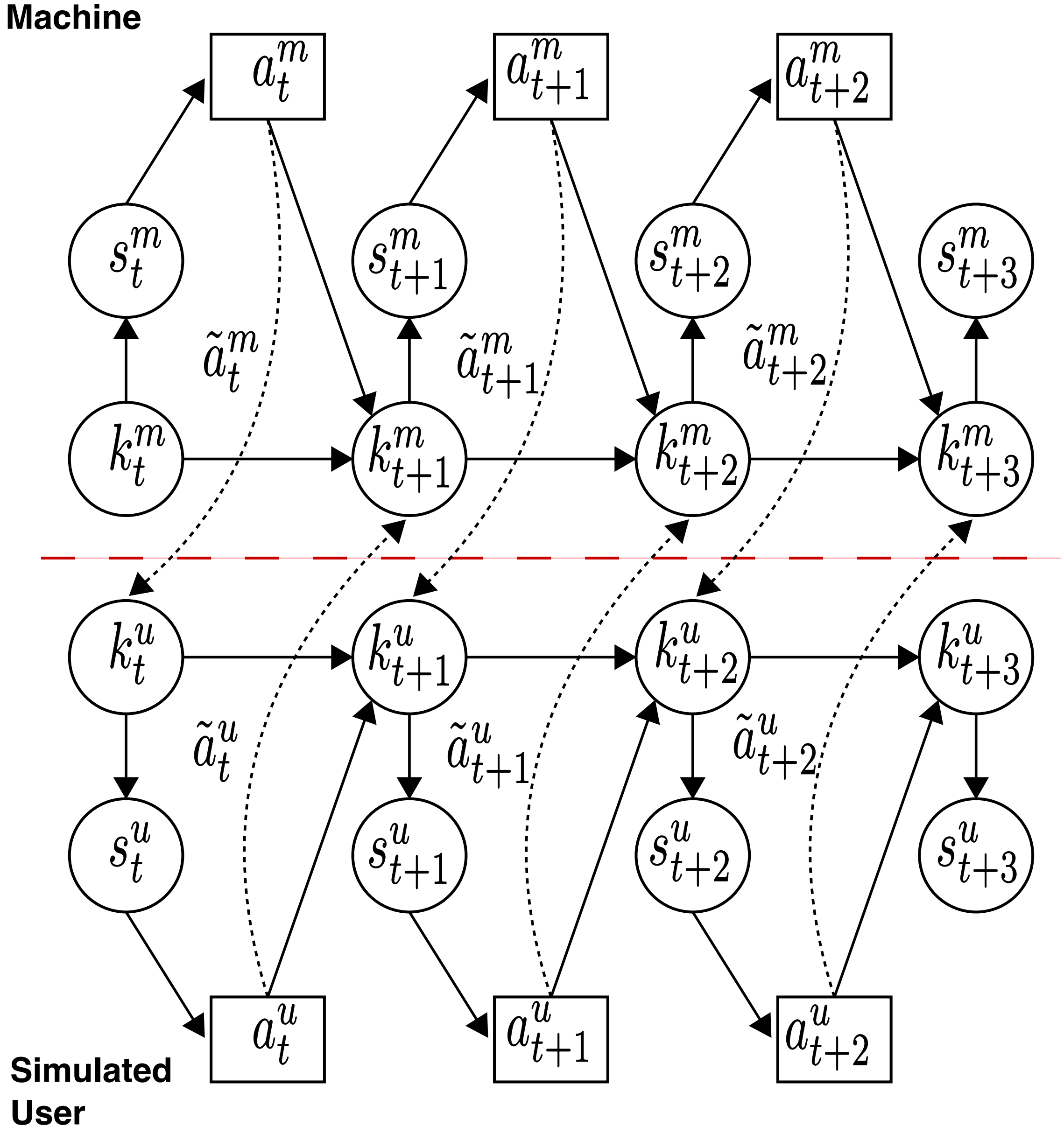


Figure 4.2: *Dynamics of human-machine communication at the dialogue act level (this diagram does not follow the conventions of dynamic Bayesian networks). A conversant in a knowledge-rich state k_t, observes a knowledge-compact state s_t, and takes dialogue act a_t in order to feed it to its knowledge-rich state and convey it to its partner, received distortedly as $\tilde{a}_t$. The current knowledge k_t, action a_t and partner response determine the next knowledge-rich state k_{t+1}, and so on until the end of the dialogue.*

Table 4.1: *Dialogue act types for task-oriented human-machine dialogues.*

Agent	ID	Dialogue Act Type	Sample Utterance
User	*pro*	provide	I want a flight from Edinburgh to London.
	rep	reprovide	I said ‘a flight to London from Edinburgh.’
	con	confirm	Yes, please.
	sil	silence	[*remain in silence*]
Machine	*req*	request	And, what is your destination city?
	apo	apology	I am sorry, I didn’t understand that.
	sic	single_IC	A flight to London.
	mic	multiple_IC	A flight from Edinburgh to London.
	sec	single_EC	I think you said London, is that correct?
	mec	multiple_EC	I heard from Paris to London, is that right?
	acc	accept_slot	[*move to next ascending slot with lowest value*]
	dbq	db_query	[*performs a database query*]
	ofr	offer	Which option would you like?
	sta	status	Please wait while I query the database.
	pre	present	The cost of this flight is 120 pounds.
	rel	relax	Try again with some different information.
	ack	acknowledgement	All right, this flight has been booked.
	ope	opening	Welcome to the travel planning system.
	clo	closing	Thank you for calling, good bye.

Abbreviations: IC = Implicit Confirmation, EC = Explicit Confirmation.

4.3.1 Knowledge representation for conversational agents

The proposed dialogue simulator uses ontologies to represent the conversant’s knowledge base. An ontology in its simplest form can be characterized as a data model with the following tuples: instances of classes, classes in the domain, attributes of classes, and relationships between classes creating a hierarchical structure that specifies how objects relate to one another, resembling the object-oriented paradigm (Gruber, 1993; Uschold and Gruninger, 1996; Staab and Studer, 2004). Tables B.1 and B.2 show the classes to instantiate for creating the knowledge bases k^m and k^u, where attributes have either deterministic or stochastic values. This chapter avoids the issue of inference and focuses on adding information with the instruction *update(k,class.attribute, value)* and querying what is known with the instruction *get(k,class.attribute)*.

4.3.2 Modelling conversational behaviour

The proposed dialogue simulation approach models the behaviour of both conversants. Whilst the simulated environment can be used to learn dialogue strategies, the machine behaviour can be used as a baseline to compare its performance against other dialogue strategies. Algorithm 5 specifies the high-level steps for simulating a task-oriented human-machine conversation. Briefly, the algorithm starts by initializing parameters for the knowledge bases of both conversants. It employs three functions described later: π_t^m is the machine dialogue strategy, π_t^u is the user dialogue strategy, and δ is the distorter of machine/user dialogue acts. A conversant interacts with their partner by: (a) observing the current knowledge-compact state, (b) selecting an appropriate dialogue act type, (c) generating a dialogue act with the current dialogue act type in context, (d) distorting the dialogue act to simulate misrecognitions or misunderstandings, (e) updating its knowledge-rich state with the undistorted dialogue act, and (f) updating the knowledge-rich state of its partner with the distorted dialogue act.

Algorithm 5 Simulator of Task-Oriented Human-Machine Conversations

function HUMANMACHINEDIALOGUESIMULATOR( )
    $k_0^m \leftarrow$ initialize machine knowledge-rich state
    $k_0^u \leftarrow$ initialize user knowledge-rich state
    $t \leftarrow$ initialize time-step to 0
    repeat
        $s_t^m \leftarrow$ observe machine dialogue state from k_t^m
        $a_t^m \leftarrow$ choose machine dialogue act type following $\pi^m(s_t^m)$
        Generate machine dialogue act $\equiv$ dialogue act type a_t^m in context
        $\tilde{a}_t^m \leftarrow$ get distorted dialogue act from $\delta(a_t^m, k_t^m)$
        Update k_t^m with a_t^m and update k_t^u with $\tilde{a}_t^m$
        $s_t^u \leftarrow$ observe user dialogue state from k_t^u
        $a_t^u \leftarrow$ choose user dialogue act type following $\pi^u(s_t^u)$
        Generate user dialogue act $\equiv$ dialogue act type a_t^u in context
        $\tilde{a}_t^u \leftarrow$ get distorted dialogue act from $\delta(a_t^u, k_t^u)$
        Update k_t^u with a^u and update k^m with $\tilde{a}_t^u$
        $t \leftarrow t+1$
    until one of the conversants terminates the conversation
end function

The process described previously, iterates until one of the conversants terminates the dialogue at T time-steps. Enumerating all possible machine or user dialogue acts usually results in large sets. Therefore, the approach taken in this chapter assumes that the action selection of both conversants is based on dialogue act types rather than dialogue acts[2]. This is beneficial because dialogue act types represent relatively small sets. Based on this, the machine takes actions following dialogue strategy π^m, and the user takes actions following dialogue strategy π^u defined by

$$\pi^m(s_t^m) = \begin{cases} ope & \text{if} \quad \text{first time step} \\ req & \text{if} \quad \text{unknown slot in focus} \\ sic+req & \text{if} \quad \text{unknown slot in focus and Single Slot to Confirm (SSC)} \\ mic+req & \text{if} \quad \text{unknown slot in focus and Multiple Slots to Confirm (MSC)} \\ apo+req & \text{if} \quad \text{slot in focus with low confidence level} \\ sec & \text{if} \quad \text{slot in focus with medium confidence level and SSC} \\ mec & \text{if} \quad \text{slot in focus with medium confidence level and MSC} \\ acc & \text{if} \quad \text{slot in focus with high confidence level} \\ dbq+sta & \text{if} \quad \text{null database result and confirmed non-terminal slots} \\ pre+ofr & \text{if} \quad \text{database result with few uninformed tuples} \\ apo+ofr & \text{if} \quad \text{terminal slot with low confidence level} \\ ofr & \text{if} \quad \text{unconfirmed terminal slot and db tuples presented before} \\ ack & \text{if} \quad \text{unacknowledged dialogue goal and confirmed terminal slot} \\ rel & \text{if} \quad \text{empty database result and confirmed non-terminal slots} \\ clo & \qquad \text{otherwise} \end{cases} \tag{4.1}$$

$$\pi^u(s_t^u) = \begin{cases} pro & \text{if} \quad \text{last machine action is a request or offer} \\ con & \text{if} \quad \text{last machine action is a correct explicit confirmation or incorrect} \\ & \qquad \text{explicit confirmation (the latter only with some probability)} \\ rep & \text{if} \quad \text{last machine action is an apology or incorrect confirmation} \\ sil & \qquad \text{otherwise} \end{cases} \tag{4.2}$$

Once an action has been chosen, it takes context[3] into account so that conversations can be generated at the dialogue act level. Although the strategy π^u may not include all possible realistic behaviours, it yields coherent behaviour, and its evaluation is addressed later. Finally, the strategy π^m is the one which we hypothesise will be outperformed by the reinforcement learning agents as described in the next chapters.

[2]Table 4.2 shows an example using dialogue acts, dialogue act types would ignore slot-value pairs.
[3]Context is given by the dialogue state, which specifies the slot in focus, slots to fill or confirm, etc.

4.3.3 Speech recognition error simulation

Due to the fact that current Automatic Speech Recognition (ASR) technology is far from perfect, errors have to be modelled in the simulated environment. For such a purpose, user dialogue acts were distorted according to

$$\delta(a_t, k_t) = \begin{cases} a_t \text{ without distortions} & \text{if} \quad p(random) \leq 1 - p(error) \\ a_t \text{ with insertions/substitutions/deletions} & \text{otherwise} \end{cases} \tag{4.3}$$

where the amount of error for each conversant is retrieved from their knowledge base as probability $p(error) = get(k_t, recognition.ker)$. As a fixed keyword error rate was assumed due to the lack of training data, errors were sampled with a flat distribution for each slot value (keyword) in the dialogue act a_t: that is, equal amounts of insertions, substitutions and deletions. Once keywords had been distorted, they were assigned the well known three-tiered confidence levels to indicate their recognition confidence. For each keyword, a confidence level was sampled from one of the distributions shown in Figure 4.3. Confidence levels were used to analyze the effects of different ASR confidence distributions. They were preferred because the true distributions of confidence scores for correct and incorrect recognition were assumed to be unknown.

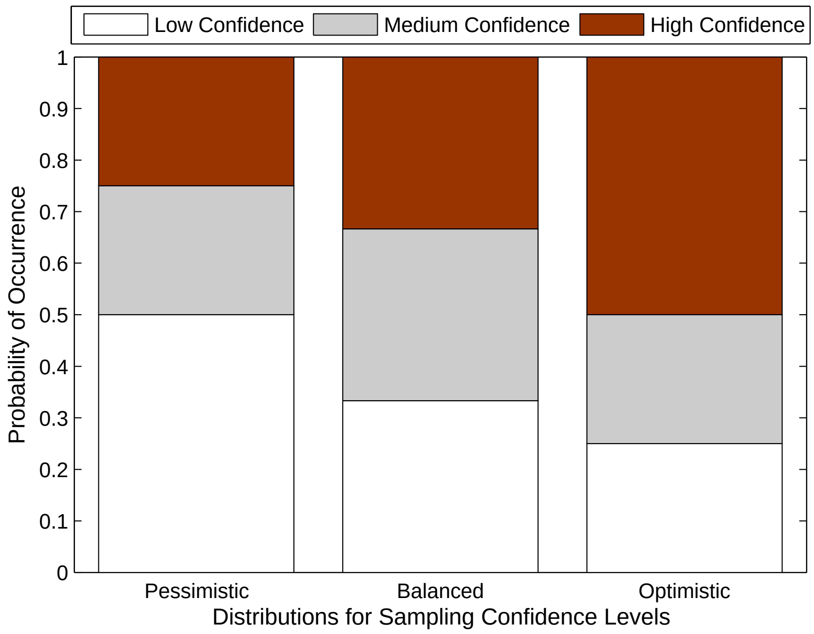


Figure 4.3: *Discrete probability distributions for sampling three-tiered speech recognition confidence levels assigned to keywords in distorted user dialogue acts* $\tilde{a}^u_t$.

4.3.4 Database querying simulation

Using a real database for dialogue simulation is simply impractical due to the lengthy time required to execute large amounts of queries. For such a purpose, a model to simulate database queries is much more practical for fast simulations, which are due to computations in main memory rather than in secondary memory. In this way, the proposed simulation environment produced the database outcomes according to

$$d(s_t^m) = \begin{cases} \text{null} & \text{if} \quad \text{slots in the dialogue state } s_t^m \text{ are unfilled} \\ \text{none} & \text{if} \quad \text{non-terminal slots in } s_t^m \text{ are confirmed and } p(random) \leq 0.1 \\ \text{few} & \text{if} \quad \text{non-terminal slots in } s_t^m \text{ are confirmed} \\ \text{many} & \quad\quad \text{otherwise.} \end{cases} \tag{4.4}$$

4.4 Experimental spoken dialogue systems

4.4.1 Case study: flight booking system

This is a 6-slot mixed-initiative spoken dialogue system in the flight booking domain, used as an example in the previous chapter. Its state representation is described in Table B.3, which shows the state variables that represent the machine knowledge-compact states s_t^m; see Table B.2 for more details of the state variables. For action selection, Table B.4 shows the action set based on dialogue act types, which correspond to dialogue acts when they take context into account. This dialogue system had 281250 states and 10 actions, resulting in 2.8 million state-actions (see equation 4.5). If V is the set of state variables and A is the action set, then:

$$|S \times A| = \Big(\prod_{v_i \in V} |v_i|\Big) \times |A|. \tag{4.5}$$

Using this state representation, action set, and simulated behaviours described in the previous section, simulated conversations at the dialogue act level can be generated. A sample simulated conversation is shown in Table 4.2, which gives an example of coherent simulated user behaviour. In this sample dialogue, due to a machine misunderstanding of ‘destination city’ in the second system turn, the simulated user reprovides its value despite the fact that is being requested for a different slot. Between system and user turns, dialogue acts are distorted as described in section 4.3. Arguably, this is the kind of behaviour that is necessary to generate realistic dialogue simulations.

Table 4.2: *Sample dialogue in the flight booking system. Although simulations are only based on dialogue acts, an equivalent wording is given for a better understanding. This dialogue shows a sample speech recognition error after the first user utterance.*

Agent	Dialogue Act	Wording
SYS	req(dep_city)	Tell me your flight information.
USR	pro(dep_city=edinburgh, des_city=paris)	*I would like a flight from Edinburgh to Paris.*
ASR		*[I would like a flight from Edinburgh to Rome]*
SYS	mic(dep_city=edinburgh, des_city=rome)+req(date)	OK, a flight from Edinburgh to Rome, when do you want to travel?
USR	rep(des_city=paris)	*I said 'to Paris.'*
SYS	sic(des_city=paris) +req(date)	OK, a flight to Paris, when do you want to fly?
USR	pro(date=01dec, airline=british_airways)	*First of December, flying with British Airways.*
SYS	apo(date)+req(date)	I'm sorry, when do you want to travel?
USR	pro(date=01dec)	*On the first of December.*
SYS	mic(date=01dec, airline= british_airways)+req(time)	Travelling on the first of December with British Airways, at what time do you prefer your flight?
USR	pro(time=morning)	*In the morning.*
SYS	apo(time)+req(time)	I'm sorry, at what time do you prefer your flight?
USR	rep(time=morning)	*In the morning.*
SYS	sec(time=morning)	Did you say in the morning?
USR	con(time=yes)	*Yes please.*
SYS	dbq(db_query)+sta(db)	Please wait while I retrieve information ...
USR		
SYS	pre(db_result)+ofr(flight)	All right, flights from Edinburgh to Paris on the first of December in the morning travelling with British Airways: choice1, choice2, choice3. Which choice do you prefer?
USR	pro(flight=choice2)	*The second one please.*
SYS	sec(flight=choice2)	Did you say choice two?
USR	con(flight=yes)	*That's right.*

Note: more complex information presentation is considered beyond the scope of this work.

4.4.2 Case study: travel planning system

This is a 26-slot multi-goal mixed-initiative spoken dialogue system in the travel planning domain, allowing users to book single flights, return flights, hotels and cars. This system is similar to the CMU Communicator (Rudnicky et al., 1999), part of the DARPA Communicator systems (Walker et al., 2001). It supports the following features: hand-crafted or learnt dialogue strategies, multiple goals within a single dialogue (see Table B.5), and implicit switching across flight dialogue goals. The action and state spaces are described in Tables B.6 and B.7. This dialogue system had 4.5×10^{22} states and 15 actions, resulting in 6.7×10^{23} state-actions (see equation 4.5). Such a state-action space (decomposed as in the next chapter), and simulated behaviours of the previous section, can be used to simulate conversations at the dialogue act level. A sample simulated dialogue is shown in Tables 4.3 and 4.4, which gives an example of switching across dialogue goals. The scenario is as follows: the user aims to book a return flight, but due to finding an empty database result, the user is asked to try with some different information (possibly switching from return flight to outbound flight and back again), once the system returns where it was, it continues the dialogue.

4.5 Evaluating user and machine dialogue behaviour

The evaluation metrics described in this section have two purposes: (1) to investigate if simulated user behaviour generates user responses that resemble human responses, and (2) to investigate if the hand-crafted dialogue strategy described in this chapter is a reasonable baseline to compare against other competing dialogue strategies. This section describes metrics for such a purpose, but due to the fact that they require real annotated dialogue data, their evaluation is reported in chapter 7.

4.5.1 Evaluation metrics for user behaviour

The evaluation of real and simulated user behaviour is carried out with three evaluation metrics from two different perspectives. The first perspective evaluates **dialogue similarity**, where *precision-recall* quantifies the predicted user responses from a user simulation model (strongly penalizes unseen responses (Schatzmann et al., 2005a)), and *probabilistic dialogue similarity* complements the previous metric by comparing probability distributions of user dialogue acts based on the Kulback-Leibler divergence (Cuayáhuitl et al., 2005). The second perspective evaluates **dialogue coherence**, where

Table 4.3: *Sample simulated dialogue in the travel planning system (part one).*

Agent	Dialogue Act	Wording
SYS	ope()	Welcome to the travel planning dialogue system.
SYS	req(dep_city)	Tell me your flight information.
USR	pro(dep_city=edinburgh, des_city=london, airline=air_france, flight_type=return)	*I would like a return flight from Edinburgh to London travelling with Air France.*
SYS	mic(dep_city=edinburgh, des_city=london,airline=lufthansa, flight_type=return)+req(dep_date)	A return flight from Edinburgh to London travelling with Lufthansa. When would you like to fly?
USR	rep(airline=air_france, dep_date=eighteenth_november, dep_time=afternoon)	*I want to travel with Air France on the eighteenth of November in the afternoon.*
SYS	mec(airline=air_france dep_date=eighteenth_november, dep_time=afternoon)	Did you say on the eighteenth of November in the afternoon travelling with Air France?
USR	con(dep_date=yes)	*Yes*
SYS	req(ret_date)	When do you want to return?
USR	pro(ret_time=evening, ret_date=twenty_first_february)	*In the evening of the twenty first of February.*
SYS	mec(ret_time=evening, ret_date=twenty_first_february)	Did you say on February twenty first in the evening?
USR	con(ret_time=yes,ret_date=yes)	*That's correct.*
SYS	dbq(db)+sta(db)	Let me retrieve your information.
SYS	rel(return_flight)	I'm sorry, I didn't find any flights from ... Please try again with some different information.
USR	rep(airline=british_airways)	*Try flying with British Airways.*
SYS	sec(airline=british_airways)	Did you say travelling with British Airways?
USR	con(airline=yes)	*Yes I did.*
SYS	dbq(db)+sta(db)	Let me retrieve your information.
SYS	pre(db)+ofr(choice)	I found 2 flights ... Which choice do you prefer?
USR	pro(choice=choice_two)	*Number two please*
SYS	sec(choice=choice_two)	Did you say choice two?
USR	con(choice=yes)	*That's correct.*

Table 4.4: *Sample simulated dialogue in the travel planning system (part two).*

Agent	Dialogue Act	Wording
SYS	ack(return_flight)	OK, I've added this flight to your itinerary.
SYS	req(want_hotel)	Do you want a hotel in London?
USR	pro(want_hotel=yes)	*Yes please.*
SYS	req(location)	Tell me your hotel information.
USR	pro(location=downtown, price=cheap,brand=any_brand)	*I would like a cheap room in the city centre with any hotel.*
SYS	mec(location=downtown, price=cheap, brand=any_brand)	Did you say a cheap hotel in the city centre with any hotel brand?
USR	con(location=yes, price=yes,brand=yes)	*Yes I did*
SYS	dbq(db)+sta(db)	Let me retrieve your information.
SYS	pre(db)+ofr(choice)	I found three cheap hotels ... Which choice do you prefer?
USR	pro(choice=choice_one)	*Choice one please*
SYS	sec(choice=choice_one)	Did you say choice one?
USR	con(hoice=yes)	*That's right.*
SYS	ack(hotel_booking)	OK, I've added this hotel to your itinerary.
SYS	req(want_car)	Would you like a car in London?
USR	pro(want_car=yes)	*Yes please.*
SYS	req(car_type)	What kind of car would you like?
USR	pro(car_type=compact_car, location=airport)	*I would like a compact car near the airport.*
...	...	...
...	...	...
SYS	req(summary)	Do you want a summary of your trip?
USR	pro(summary=yes)	*Yes please*
SYS	dbq(db)+sta(db)	Let me retrieve your information.
SYS	pre(db)+ofr(book_trip)	All right, you have a flight leaving ... Do you want to book this trip?
USR	pro(book_trip=yes)	*Yes please.*
SYS	ack(summarize_trip)	All right, your trip has been booked.
SYS	clo()	Thanks for calling the travel planning system.

coherence error rate ignores the seen or unseen user responses, instead, it classifies them into coherent or incoherent responses.

To illustrate the similarity and coherence metrics consider the sub-dialogues below with common system responses assumed from logged real data, but user responses may be real (see an example in Table 4.5) or simulated (see examples in Tables 4.6 and 4.7). The acronyms of dialogue act types are described in Table 4.1.

Table 4.5: *Sample sub-dialogue with user responses assumed from logged real data.*

Agent	Dialogue Act	Wording
SYS	gre(),	Welcome to the travel planning system.
	req(dep_city)	Tell me your flight information.
USR	pro(dep_city=amsterdam,	*I would like a return flight leaving from*
	flight_type=return)	*Amsterdam.*
SYS	sic(flight_type=return),req(dep_city)	A return flight, where are you leaving from?
USR	pro(dep_city=amsterdam)	*Amsterdam*

Table 4.6: *Sample sub-dialogue with simulated coherent user responses.*

Agent	Dialogue Act	Wording
SYS	gre(),	Welcome to the travel planning system.
	req(dep_city)	Tell me your flight information.
USR	pro(dep_city=paris,dep_time=morning,	*A return flight from Paris travelling in the*
	airline=air france,flight_type=return)	*morning with Air France*
SYS	sic(flight_type=return),req(dep_city)	A return flight, where are you leaving from?
USR	pro(dep_city=amsterdam)	*Amsterdam*

Table 4.7: *Sample sub-dialogue with simulated random user responses.*

Agent	Dialogue Act	Wording
SYS	gre(),	Welcome to the travel planning system.
	req(dep_city)	Tell me your flight information.
USR	con(dest_city=yes)	*Yes*
SYS	sic(flight_type=return),req(dep_city)	A return flight, where are you leaving from?
USR	pro(des_city=paris)	*To Paris*

4.5.1.1 Precision-Recall

This measure is commonly used in the information retrieval field, and was suggested by (Schatzmann et al., 2005a) to evaluate how well a user simulation model can predict real user dialogue behaviour. *Precision* specifies the fraction of correctly predicted real user responses from all simulated responses. *Recall* specifies the fraction of correctly predicted real user responses from all real responses. They are expressed as

$$\text{Precision} = \frac{\text{Number of correctly predicted user responses}}{\text{Total number of simulated user responses}}, \tag{4.6}$$

and

$$\text{Recall} = \frac{\text{Number of correctly predicted user responses}}{\text{Total number of real user responses}}. \tag{4.7}$$

These scores are interpreted as the higher the more realistic the user responses. An average score of recall (R) and precision (P) called *F-measure* is defined by

$$F = \frac{2PR}{(P+R)}. \tag{4.8}$$

If we want to compute the F-measure score in dialogue data, the slot values can be ignored to reduce data sparsity while preserving the conveyed information. (Schatzmann et al., 2005a) suggested to compute precision-recall by considering a user dialogue act as a sequence of actions, e.g. the dialogue act 'pro(dep_city,flight_type)' is equivalent to $\{$pro(dep_city), pro(flight_type)$\}$. Considering the given sample sub-dialogues, the F-measure score for real vs simulated coherent responses is $F = 0.75$, and the score for real vs simulated random responses is $F = 0$. Alternatively, the scores can be computed in a more strict way by considering each user response as a single user action instead of multiple ones. Precision-recall can be recomputed as follows: the scores for real vs simulated coherent responses are $F = 0.5$; and the score for real vs simulated random responses is $F = 0$.

4.5.1.2 Probabilistic Dialogue Similarity

The purpose of this measure is to evaluate the probabilistic similarity between two sets of dialogues. The similarity between real and simulated dialogues has been analyzed using the Kulback-Leibler divergence (Cuayáhuitl et al., 2005), and here we propose to apply it in a simpler way. First, compute two smoothed probability distributions of machine-user dialogue acts, without slot values for reduced combinations: P for one data set and Q for the other. For example: P represents a distribution of the set of real dialogues and Q a distribution of the set of simulated ones. Then compute the symmetric distance according to

$$D(P,Q) = \frac{D_{KL}(P \parallel Q) + D_{KL}(Q \parallel P)}{2}, \tag{4.9}$$

where D_{KL} is the Kulback-Leibler divergence (distance) between P and Q:

$$D_{KL}(P \parallel Q) = \sum_i p_i \, log_2 \, (\frac{p_i}{q_i}). \tag{4.10}$$

Tables 4.8 and 4.9 use the sample sub-dialogues of this subsection in order to show the divergence between real and simulated coherent user responses, and between real and simulated random user responses. The probability distributions of occurrence P and Q were smoothed by assigning a probability mass of 0.1 to unseen events, and the method of preference can be used to address the issue of data sparsity. It can be observed that the symmetric divergence between real and simulated random user responses (2.536) is greater than between real and simulated coherent ones (0.759). This reflects the intuitive perception that the more realistic the user responses, the shorter the divergence.

Table 4.8: *Dialogue similarity results for real vs simulated coherent sub-dialogues.*

Dialogue Act Pairs (SYS:USR)	P	Q	$D_{KL}(P\|\|Q)$	$D_{KL}(Q\|\|P)$
gre(),req(dep_city):pro(dep_city,flight_type)	0.45	0.45	0.000	0.000
sic(flight_type)+req(dep_city):pro(dep_city)	0.45	0.10	0.976	-0.217
sic(flight_type)+req(dep_city):pro(dep_city, des_city,dep_time,airline)	0.10	0.45	-0.217	0.976
Divergence			0.759	0.759

Table 4.9: *Dialogue similarity results for real vs simulated random sub-dialogues.*

Dialogue Act Pairs (SYS:USR)	P	Q	$D_{KL}(P\|\|Q)$	$D_{KL}(Q\|\|P)$
gre(),req(dep_city):pro(dep_city,flight_type)	0.45	0.05	1.426	-0.158
sic(flight_type)+req(dep_city):pro(dep_city)	0.45	0.05	1.426	-0.158
gre(),req(dep_city):con(des_city)	0.05	0.45	-0.158	1.426
sic(flight_type)+req(dep_city):pro(des_city)	0.05	0.45	-0.158	1.426
Divergence			2.536	2.536

It can be observed that this metric gives the same ordering on user simulations than the precision-recall metric. A validation of this ordering taking into account a corpus of real human-machine dialogues is reported on chapter 7.

4.5.1.3 Coherence Error Rate

An evaluation metric called *Coherence Error Rate (CER)* is proposed due to the fact that most previously used metrics penalize unseen user responses even when they may be realistic. The key assumption in this metric is that given a user knowledge-base k_t^u and a set of dialogue coherence rules encoded into a function, we can evaluate – in an approximated form – whether a user action a_t^u is coherent or not. This metric rates errors (in this context, incoherent dialogue acts) from a set of observed events (user dialogue acts in the data), in terms of dialogue act types (see Table 4.1):

$$CER = \frac{\sum incoherent(a_t^u, k_t^u)}{count(a_t^u)} \times 100, \tag{4.11}$$

where the coherence of user dialogue acts is evaluated according to

$$incoherent(a_t^u, k_t^u) = \begin{cases} 0 & \text{if} \quad a_t^u \in \{pro, rep\} \text{and unconfirmed slot in focus in } k_t^u \\ 0 & \text{if} \quad a_t^u \in \{con\} \text{ and } a_t^m \in \{sec, mec\} \\ 0 & \text{if} \quad a_t^u \in \{pro, rep\} \text{ and } a_t^m \in \{rel\} \\ 1 & \qquad \text{otherwise.} \end{cases} \tag{4.12}$$

Equation 4.12 is suited for simple slot-filling applications, but for more complex dialogues more rules have to be added. This metric takes into account user dialogue acts and decomposes them into dialogue acts with a single slot and without slot value, e.g. $pro(des\_city)$. This procedure incorporates the conveyed information, and assumes that the slot values are always consistent given a user goal at the beginning of the conversation. In addition, this evaluation metric considers the user dialogue act 'silence' as incoherent, the explanation for this consideration is because whatever the user said (e.g. mumbles or out-of-vocabulary words), it was not possible to extract a user dialogue act contributing to the conversation.

Given the sample sub-dialogues of this subsection, Table 4.10 shows the results of coherence for real, simulated coherent and simulated random user responses: 0%, 0%, 50%, respectively. Note that although simulated coherent user responses do not match the real ones, they are not being penalized because they are user responses that make sense according to the dialogue history.

Table 4.10: *Results of coherence for real and simulated user responses.*

Data Set	Dialogue Act Pairs (SYS:USR)	$incoherent(a_t^u, k_t^u)$
Real	gre(),req(dep_city):pro(dep_city)	0
	gre(),req(dep_city):pro(flight_type)	0
	sic(flight_type),req(dep_city):pro(dep_city)	0
Simulated coherent	gre(),req(dep_city):pro(dep_city)	0
	gre(),req(dep_city):pro(dep_time)	0
	gre(),req(dep_city):pro(airline)	0
	gre(),req(dep_city):pro(flight_type)	0
	sic(flight_type),req(dep_city):pro(dep_city)	0
Simulated random	gre(),req(dep_city):con(dest_city)	1
	sic(flight_type),req(dep_city):pro(des_city)	0

4.5.2 A reasonable choice of baseline machine behaviour

The use of speech recognition confidence scores has forced spoken dialogue strategies to handle tradeoffs among acceptance, confirmation and rejection events e_i, which can be classified as correct $E^c = \{ca, cc, cr\}$ or incorrect $E^f = \{fa, fc, fr\}$. Table 4.11 shows the categories of recognition events. A reasonable dialogue strategy would choose actions maximizing correct acceptance/confirmation/rejection events, whilst minimizing the incorrect ones. A simple metric to quantify these events is referred to as an *Event Error Rate (EvER)*:

$$EvER = \frac{count(e_i \in E^f)}{count(e_j \in \{E^c, E^f\})} \times 100. \tag{4.13}$$

Table 4.11: *Speech recognition events in spoken dialogue systems.*

Recognition Event	Correct	False (or incorrect)
Acceptance	ca	fa
Confirmation	cc	fc
Rejection	cr	fr

For such a purpose, consider that speech recognition hypotheses fall within three equally distributed regions of confidence scores (assuming no training data): low confidence, medium confidence, and high confidence. A sample dialogue annotated with speech recognition events is shown in figure 4.12.

Table 4.12: *Sample dialogue in the flight booking domain annotated with speech recognition events, showing an EvER score of* 33% *(including 3 incorrect events and six correct ones). Notation: ca=correct acceptance, cc=correct confirmation, cr=correct rejection, fa=false acceptance, fc=false confirmation, fr=false rejection.*

Agent	Wording	Speech Recognition Event
SYS USR	Tell me your flight information. *I would like a flight from Edinburgh to Paris.*	
SYS USR	OK, a flight from Edinburgh to Rome, when do you want to travel? *I said 'to Paris.'*	ca(dep_city), fa(des_city)
SYS USR	OK, a flight to Paris, when do you want to fly? *First of December, flying with British Airways.*	ca(des_city)
SYS USR	I'm sorry, when do you want to travel? *On the first of December.*	fr(date)
SYS USR	Travelling on the first of December with British Airways, at what time do you prefer your flight? *In the morning.*	ca(date),ca(airline)
SYS USR	I'm sorry, at what time do you prefer your flight? *In the morning.*	fr(time)
SYS USR	Did you say in the morning? *Yes please.*	cc(time)
SYS USR	Please wait while I retrieve information ...	
SYS USR	All right, flights from Edinburgh to Paris on the first of December in the morning travelling with British Airways: choice1, choice2, choice3. Which choice do you prefer? *The second one please.*	
SYS USR	Did you say choice two? *That's right.*	cc(flight)

In addition, consider the confirmation strategies Π of Table 4.13 for the three confidence regions. Which confirmation strategy is a better baseline of machine behaviour? For perfect speech recognizers it has to be 'Strategy1', because it leads to more efficient conversations in terms of number of system turns; but this is unrealistic. Thus, a more reasonable choice of dialogue strategy is the one obtaining the lowest EvER score, and can be expressed as

$$\text{Baseline Strategy} = \arg\min_{\pi_i \in \Pi} EvER(\pi_i). \tag{4.14}$$

Table 4.13: *Confirmation strategies for different recognition confidence score regions. Notation: IC=implicit confirmations, EC=explicit confirmations, and AP=apologies.*

Strategy	Low Confidence	Medium Confidence	High Confidence
Strategy1	IC	IC	IC
Strategy2	EC	IC	IC
Strategy3	AP	EC	IC
Strategy4	AP	EC	EC
Strategy5	EC	EC	EC

Results for this baseline strategy taking into account a corpus of real human-machine dialogues is reported on chapter 7 (page 156).

4.6 Discussion

In order to learn spoken dialogue strategies in a practical and effective way, a number of issues must be addressed in the dialogue simulation environment. The following issues highlight the strengths and weaknesses of the proposed simulation framework: (a) training data, (b) coherent user behaviour, (c) speech recognition error simulation, (d) complexity of user behaviour, and (e) evaluation of simulated behaviour.

Firstly, without training data, how can a dialogue environment be simulated? In the field of spoken dialogue systems, the chicken-and-egg problem seems unavoidable: data is required to build a system and the system is required to collect data (Zue and Glass, 2000). Besides, even in the presence of collected dialogue data, it is expensive and time-consuming to annotate for training a model that simulates the conversational environment. One possible solution is to use a heuristic approach to model the dynamics of speech-based human-machine communication. This is the approach that is taken in this chapter, and a major criticism is that it may not truly reflect real conversational behaviour. Nonetheless, its use is justified if it helps to find errors in dialogue strategies, and/or if it helps learning agents to find dialogue strategies that outperform hand-crafted machine behaviour. The latter is addressed in the following chapters.

Secondly, do real users act with coherent dialogue behaviour? If we assume that real users provide dialogue acts in a logically integrated and consistent way, then the approach of coherent behaviour is approximating real behaviour. Previous studies suggest that human dialogues maintain coherent behaviour if they interact in a joint activity (Reichman, 1978; Grosz and Sidner, 1986; Clark, 1996) where speech acts are the basis for understanding dialogue coherence (Austin, 1962; Searle, 1969). Based on this assumption, user simulation following dialogue coherence is a reasonable approach to follow, but it has received little attention in the dialogue simulation field. Therefore, it remains to be investigated if user simulation approaches taking into account coherence-based behaviour can help to optimize good dialogue policies.

Thirdly, the speech recognition error modelling in the proposed conversational simulation environment may not be very realistic. This is to be expected because the proposed approach does not assume any training data. However, the proposed simulation approach can be enhanced with probability distributions of errors estimated from real annotated data as in Schatzmann et al. (2007b). Notice that the issue of real training data is crucial for the simulation of more realistic behaviour. Due to the fact that collecting training data is costly and time consuming, a potential future research direction is to investigate methods that generalize simulated behaviours for spoken dialogue systems in different domains.

Fourthly, another criticism in the proposed conversational simulation environment is that simulated user behaviour was narrowed down to few dialogue acts (provide information, re-provide information, confirm information, and silence). Whilst this represents basic behaviour for interacting with information seeking dialogue systems, richer dialogue behaviour must be taken into account such as asking questions.

Fifthly, how can simulated user behaviour be evaluated? Because there is a variety of proposals on how to evaluate user simulations, this chapter proposed two metrics to evaluate user behaviour based on dialogue coherence and similarity, and also suggested to validate their results with the more established metric ‘Precision-Recall’. On the one hand, dialogue coherence can be used to evaluate whether user actions are coherent or not, based on knowledge about the conversation with a partner conversant. On the other hand, it is complemented by dialogue similarity in order to determine how closely simulated dialogues resemble real ones in terms of machine-user pairs of dialogue acts.

Finally, what is a reasonable baseline of machine dialogue behaviour? If a simulated environment can help conversational agents to find optimal dialogue behaviours, then they need a baseline for performance comparison. However, evaluating dialogue behaviours is a difficult task despite the existence of well accepted metrics such as task completion, average system turns per dialogue, and word error rate. Nevertheless, the proposed simulation framework considered the following three-tiered confirmation strategy as a baseline: rejection of keywords with low confidence scores, explicit confirmation for medium confidence scores, and implicit confirmation for high confidence scores. Such a strategy included in equation 4.1, was used as a baseline of learnt dialogue behaviours described in the following chapters.

4.7 Summary

In this chapter a simple conversational simulation environment was proposed based on the heuristics of the dynamics of human-machine communication at the dialogue act level. This simulation environment does not require training data, generates coherent and coherent-distorted user behaviour, and is straightforward to implement and modify. The simulation environment encapsulates the following simulators: user behaviour, speech recognition error modelling and database behaviour. Included is a baseline of machine dialogue behaviour with which to compare the performance of learnt dialogue strategies. In addition, three simulation evaluation metrics under two different perspectives were described: *dialogue similarity* using ‘Precision-Recall’ and ‘the Kulback-Leibler divergence’, and *dialogue coherence* using ‘Coherence Error Rate’. These metrics require annotated real conversations at the dialogue act level. Whilst Precision-Recall is part of the state of the art in the field, the other two metrics were proposed for additional assessments of dialogue realism. The hypotheses of this chapter are three-fold:

(1) the proposed simulation environment can help learning agents to find behaviours with superior performance to hand-crafted ones,

(2) the proposed heuristic machine dialogue behaviour is a reasonable baseline, and

(3) the proposed simulation metrics can be used to evaluate dialogue realism.

Experimental results on real human-machine spoken dialogues to validate these hypotheses are reported in chapter 7.

Chapter 5

Hierarchical dialogue optimization: a divide-and-conquer approach

This chapter describes a novel approach for scalable optimization of spoken dialogue strategies using Semi-Markov decision processes and hierarchical reinforcement learning. Section 5.2 treats the optimization of machine dialogue behaviour as a Semi-Markov Decision Process (SMDP), and explains how to apply SMDPs to spoken dialogue management. Section 5.3 describes a learning algorithm for hierarchical SMDPs. Sections 5.4 and 5.5 report experiments and results using a 6-slot mixed-initiative flight booking dialogue system and a 26-slot multi-goal mixed-initiative travel planning dialogue system. Section 5.6 discusses the strengths and weaknesses of the proposed approach. The last section summarizes the chapter and draws conclusions.

5.1 Introduction

Previous investigations in the literature of spoken dialogue systems have formulated the task of dialogue strategy design as a Markov Decision Process (MDP) (Levin and Pieraccini, 1997; Levin et al., 2000) or as a Partially Observable MDP (POMDP) (Roy et al., 2000; Young, 2002; Williams, 2006), where the goal is to infer the best action for each state or belief state. The MDP and POMDP formalisms share a common problem that affects their practical application – *the curse of dimensionality*. Consequently, only small-scale systems can be optimized. This research addresses the problem of scalable dialogue optimization with hierarchical structures, optimizing sub-dialogues instead of full dialogues. A *hierarchical reinforcement learning agent* is used to provide a hierarchy of sub-solutions and behaves by executing composite and primitive actions.

5.1.1 Background on dialogue strategy learning

A human-machine dialogue can be defined as a finite sequence of information units conveyed between conversants, where the information can be described at different levels of communication such as speech signals, words, and dialogue acts. Figure 5.1 illustrates a model of human-machine communication. An interaction between both conversants can be briefly described as follows: the machine receives a distorted user speech signal $\tilde{x}_t^u$ from which it extracts a user dialogue act $\tilde{a}_t^u$ and enters it into its knowledge base; then the machine updates its dialogue state s_t^m with information extracted from its knowledge base; this dialogue state is received by the spoken dialogue manager in order to choose a machine dialogue act a_t^m, which is received by the response generation module to generate the corresponding machine speech signal conveyed to the user.

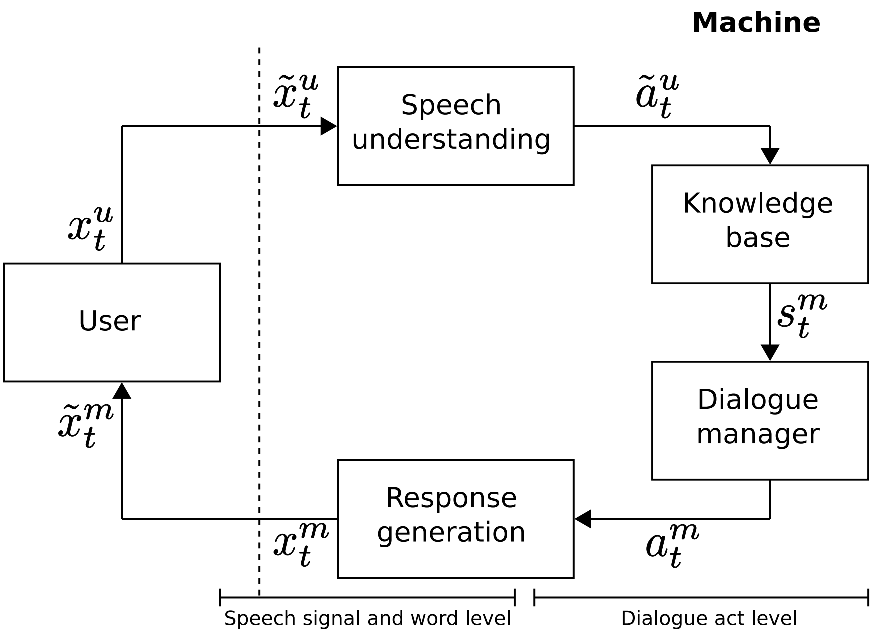


Figure 5.1: *A pipeline model of speech-based human-machine communication, where dialogue state s_t^m is used by the dialogue manager to choose action a_t^m. For dialogue strategy learning the speech signals and words can be omitted.*

A conversation follows the sequence of interactions above in an iterative process between both conversants until one of them terminates it. Assuming that the machine receives the reward r_{t+1} for executing action $a_t = a_t^m$ when the conversational environment makes a transition from state $s_t = s_t^m$ to state $s_{t+1} = s_{t+1}^m$, a dialogue can be expressed as

$$D = \{s_1, a_1, r_2, s_2, a_2, r_3, ..., s_{T-1}, a_{T-1}, r_T, s_T\}, \tag{5.1}$$

where T is the final time step. Such sequences can be used by a reinforcement learning agent to optimize the machine's dialogue behaviour (Levin and Pieraccini, 1997; Levin et al., 1998, 2000). Although human-machine conversations can be used for optimizing dialogue behaviour, a more common practice is to use simulations (see chapter 4).

A reinforcement learning dialogue agent aims to learn its behaviour from interaction with an environment, where situations are mapped to actions by maximizing a long-term reward signal (see section 2.2 for an introduction to reinforcement learning). Briefly, the standard reinforcement learning paradigm works by using the formalism of Markov Decision Processes (MDPs) (Kaelbling et al., 1996; Sutton and Barto, 1998; Bertsekas and Tsitsiklis, 1996; Boutilier et al., 1999). An MDP is characterized by a set of states S, a set of actions A, a state transition function, and a reward or performance function that rewards the agent for each selected action. Solving the MDP means finding a mapping from states to actions corresponding to

$$\pi^*(s_t) = \arg\max_{a_t \in A} Q^*(s_t, a_t), \tag{5.2}$$

where the Q function specifies the cumulative rewards for each state-action pair. An alternative for sequential decision-making under uncertainty is the POMDP model. In a POMDP the dialogue state is not known with certainty (as opposed to an MDP), and solving it means finding a mapping from belief states to actions.

Spoken dialogue systems that learn to optimize their behaviour have largely been investigated within the flat tabular reinforcement learning paradigm (Levin et al., 2000; Walker, 2000; Young, 2000; Singh et al., 2002; Scheffler, 2002; Pietquin, 2004; Williams, 2006; Young et al., 2007). The scalability of this approach is limited because search spaces grow exponentially according to the number of state variables taken into account (referred to as 'the curse of dimensionality'). Even systems with simple state representations may have large search spaces with quick growth towards intractability. This problem has led to the use of function approximation (Denecke et al., 2004; Henderson et al., 2005, 2008) in order to find solutions on reduced state-action spaces. Evolutionary methods have also been been used to find optimal dialogue policies on compact state-action spaces (Toney, 2007). All these investigations have been applied to small-scale dialogue systems aiming for a single global solution. However, little attention has been paid to finding solutions with the divide-and-conquer approach, where hierarchical POMDPs with a bottom-up approach have been applied to small state-action spaces (Pineau, 2004), and hierarchical reactive planning and learning has been used for dialogue systems with few slots of information (Lemon et al., 2006b).

5.1.2 Related work on hierarchical reinforcement learning

Prior work in the literature of artificial intelligence has investigated divide-and-conquer approaches to address the problem of reinforcement learning on large search spaces, referred to as Hierarchical Reinforcement Learning (HRL) (Watkins, 1989; Singh, 1992; Kaelbling, 1993; Dayan and Hinton, 1992; Bradtke and Duff, 1994; Karlsson, 1997; Parr, 1998; Sutton et al., 1999; Precup, 2000; Dietterich, 2000a; Ryan, 2002; Andre, 2003; Hengst, 2003; Mahadevan et al., 2004; Marthi, 2006; Ghavamzadeh and Mahadevan, 2007; Jonsson, 2008). The fundamental theory behind HRL is based on Semi-Markov Decision Processes (SMDPs) (Barto and Mahadevan, 2003), see chapter 3 for an introduction. HRL is attractive due to the following benefits: (a) improved exploration, because exploration can take multi-time steps by using low-level and high-level actions; (b) reduced computational demands, because breaking a problem into sub-problems helps to avoid irrelevant features of the flat environment state; and (c) knowledge transfer, because components of solutions learnt on previous problems can be reused in new problems. However, the price to pay for such benefits is that HRL methods may learn sub-optimal solutions. Nevertheless, HRL methods learn the best policies according to the constraints specified in the hierarchy (Dietterich, 2000a).

Related work on SMDPs and HRL can be broadly classified into two approaches: those that learn on a single SMDP and those that learn on multiple SMDPs. Methods learning on a single SMDP have focused on high-level and low-level actions to accelerate learning (Bradtke and Duff, 1994; Parr and Russell, 1997; Sutton et al., 1999; Andre and Russell, 2000). Although this approach can mitigate the curse of dimensionality problem, it is limited because the environment is represented by flat states rather than hierarchical states. Therefore, learning using a single SMDP lacks scalability and reusability. In contrast, learning on multiple SMDPs can employ hierarchical states, actions and rewards. Using hierarchical SMDPs facilitates state abstraction, meaning that smaller solutions can be found faster, with reduced computational demands, and with opportunities for policy reuse (Dayan and Hinton, 1992; Dietterich, 2000a).

This chapter investigates how to create hierarchical dialogue controllers for large MDPs. For such a purpose, it proposes to decompose a large MDP into a hierarchy of Semi-Markov Decision Processes (SMDPs), and to find the policy for each SMDP with hierarchical reinforcement learning. This approach has not been applied before to dialogue strategy learning, and it will be shown that the proposed approach is promising for efficiently optimizing the dialogue behaviour of large state-action spaces.

5.2 Dialogue as a Semi-Markov Decision Process

This thesis treats spoken dialogue control as a discrete Semi-Markov Decision Process (SMDP) in order to address the problem of scalable dialogue optimization. A discrete-time SMDP $M = <S, A, T, R>$ is characterized by a set of states S; a set of actions A; a transition function T that specifies the next state s' given the current state s and action a with probability $P(s', \tau | s, a)$; and a reward function $R(s', \tau | s, a)$ that specifies the reward given to the agent for choosing action a when the environment makes a transition from state s to state s'. The random variable τ denotes the number of time-steps taken to execute action a in state s. This formulation, based on (Dietterich, 2000a) differs from the original formulation of SMDPs (Howard, 1971; Putterman, 1994), see section 3.3 for more details. The SMDP model allows temporal abstraction, where actions take a variable amount of time to complete their execution. In this model two types of actions can be distinguished: (a) single-step actions roughly corresponding to dialogue acts, and (b) multi-step actions corresponding to sub-dialogues. Figure 5.2 illustrates a conceptual dialogue at runtime with dialogue states s_t, actions a_t and rewards r_t. Whilst the full dialogue and child dialogue execute primitive and composite actions, the grandchildren dialogues execute only primitive actions. Note that the execution of primitive actions yields single rewards and the execution of composite actions lasting τ time steps yields cumulative discounted rewards given at time $t + \tau$.

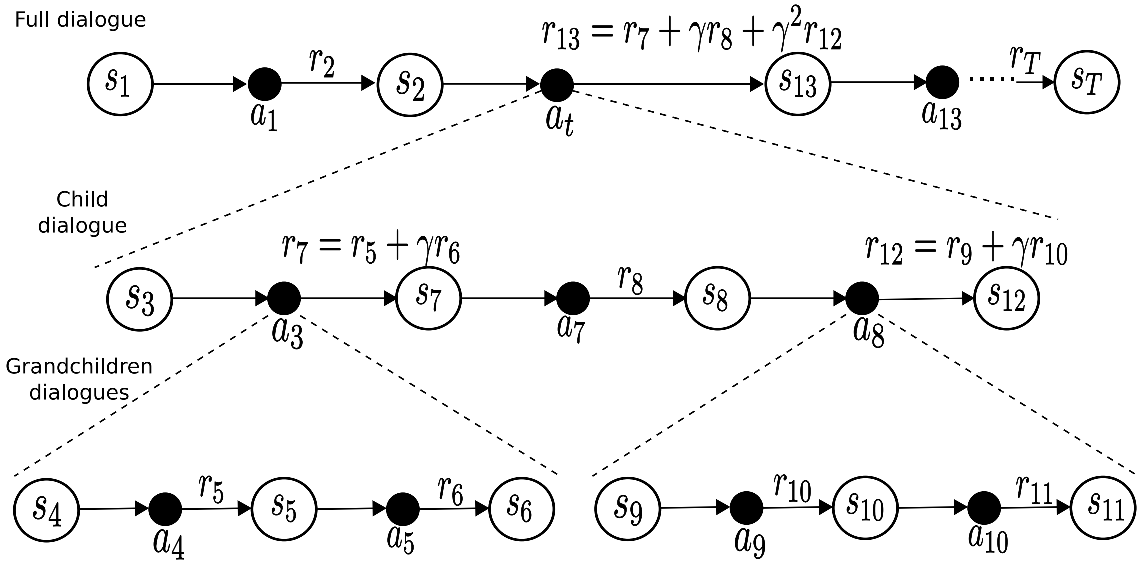


Figure 5.2: *Conceptual hierarchical dialogue at runtime with states s_t, actions a_t (lasting τ time steps) and rewards $r_{t+\tau}$. Actions a_t can be either primitive or composite, the former yield single rewards and the latter yield cumulative discounted rewards.*

5.2.1 Dialogue control using hierarchical SMDPs

This research treats each composite dialogue action as a separate SMDP as described in (Cuayáhuitl et al., 2007). In this way an MDP can be decomposed into multiple SMDPs hierarchically organized into L levels and N models per level, denoted as $\mathcal{M} = \{M^i_j\}$, where $j \in \{0,...,N-1\}$ and $i \in \{0,...,L-1\}$. Thus, any given SMDP in the hierarchy is denoted as $M^i_j = <S^i_j, A^i_j, T^i_j, R^i_j>$, see Figure 5.3 for an illustration.

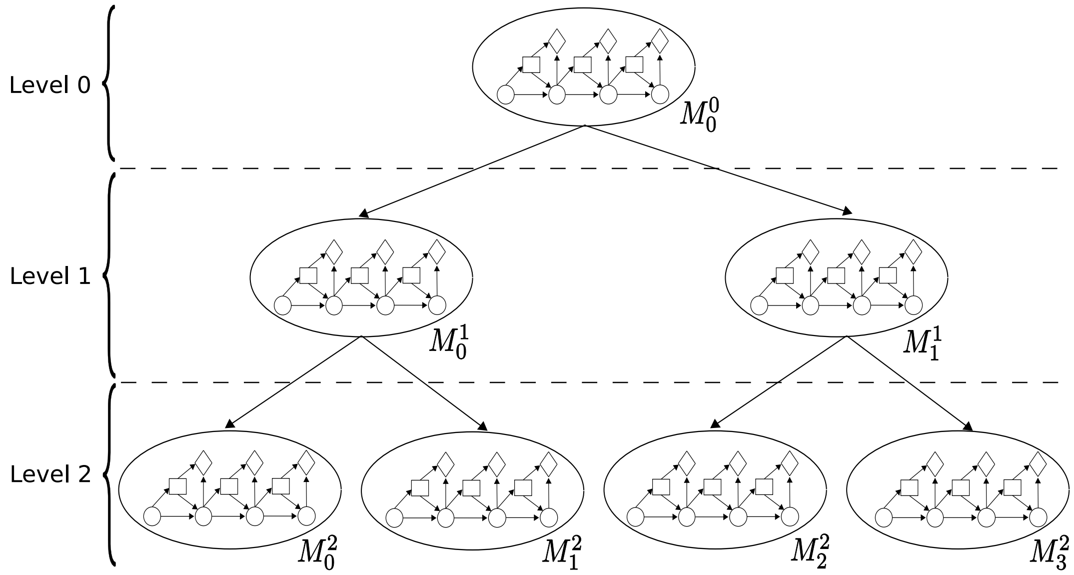


Figure 5.3: *Hierarchy of SMDPs M^i_j, where i denotes a level and j the model per level.*

The goal in an SMDP is to find an optimal policy π^*, that maximizes the reward of each visited state. The optimal value function $V^*(s)$ specifies the expected cumulative reward of state s under π^*. Similarly, the optimal action-value function $Q^*(s,a)$ specifies the expected cumulative reward for executing action a in s and then following π^*. The Bellman equations for V^* and Q^* of subtask M^i_j can be expressed as

$$V^{*i}_j(s) = \max_a \left[\sum_{s',\tau} P^i_j(s',\tau|s,a)[R^i_j(s',\tau|s,a) + \gamma^\tau V^{*i}_j(s')] \right], \tag{5.3}$$

$$Q^{*i}_j(s,a) = \sum_{s',\tau} P^i_j(s',\tau|s,a)[R^i_j(s',\tau|s,a) + \gamma^\tau \max_{a'} Q^{*i}_j(s',a')], \tag{5.4}$$

where the discount rate $0 \le \gamma \le 1$ makes future rewards less valuable than immediate rewards as it approaches 0. Finally, the optimal policy for each subtask is defined by

$$\pi^{*i}_j(s) = \arg\max_{a \in A^i_j} Q^{*i}_j(s,a). \tag{5.5}$$

These policies can be found by dynamic programming or reinforcement learning algorithms for SMDPs, the latter are preferred (see sub-section 2.2.2).

5.2.2 Decomposing a spoken dialogue manager into subtasks

Due to the fact that the process of automatically breaking an MDP into sub-problems is challenging, a heuristic approach is proposed to divide a dialogue-based MDP into a hierarchy of dialogue-based SMDPs, and to perform state abstraction in each SMDP. The heuristic decomposition described here aims to be a guideline for specifying the hierarchy of subtasks in hierarchical dialogue optimization.

5.2.2.1 Hierarchical subtask decomposition

A dialogue task is decomposed into a root subtask M_0^0 and set of meta-dialogue goals $M^1 = \{M_0^1, \ldots, M_{W-1}^1\}$. Each meta-dialogue goal is decomposed into a set of dialogue goals $M^2 = \{M_0^2, \ldots, M_{X-1}^2\}$. Then, each dialogue goal is decomposed into a set of slot filling strategies $M^3 = \{M_0^3, \ldots, M_{Y-1}^3\}$ such as for the initial slot, mandatory slots, optional slots, and terminal slot. Finally, the last stage decomposes every slot filling strategy into a set of initiative strategies $M^4 = \{M_1^4, \ldots, M_{Z-1}^4\}$ such as system-initiative and mixed-initiative. Therefore, each dialogue subtask in the hierarchy is represented with an SMDP, and the hierarchy can be denoted by $\mathcal{M} = \{M_j^i\}$. The global decomposition can have a maximum number of subtasks $|\mathcal{M}| = 1 + W + WX + WXY + WXYZ$. Finding the best hierarchy for a given conversational agent is beyond the scope of this thesis (though see (Hengst, 2003) for an approach in hierarchy discovery).

5.2.2.2 State abstraction

The decomposition above only specifies a hierarchy of dialogue subtasks, but it does not specify how to represent states with a more compact representation. This is important because the states in each subtask may have a large number of state variables, and some of them may be irrelevant for decision-making (this is also referred to as 'state abstraction'). In this thesis state abstractions are specified by the system developer. Previous work has proposed methods for automatic state abstraction, but it has been investigated for tasks with few state variables (Dietterich, 2000a; Andre and Russell, 2002; Uther, 2002; Jong and Stone, 2005; Marthi et al., 2006; Jonsson, 2008).

A bottom-up procedure was used for state abstraction in each dialogue subtask: (1) by removing irrelevant state variables such as the variables only relevant for other subtasks; and (2) by clustering state variables from child subtasks, e.g. a set of slots in a semantic frame can be described with a single variable in the parent subtask. Figure 5.4 shows this procedure aiming to represent the dialogue state more compactly.

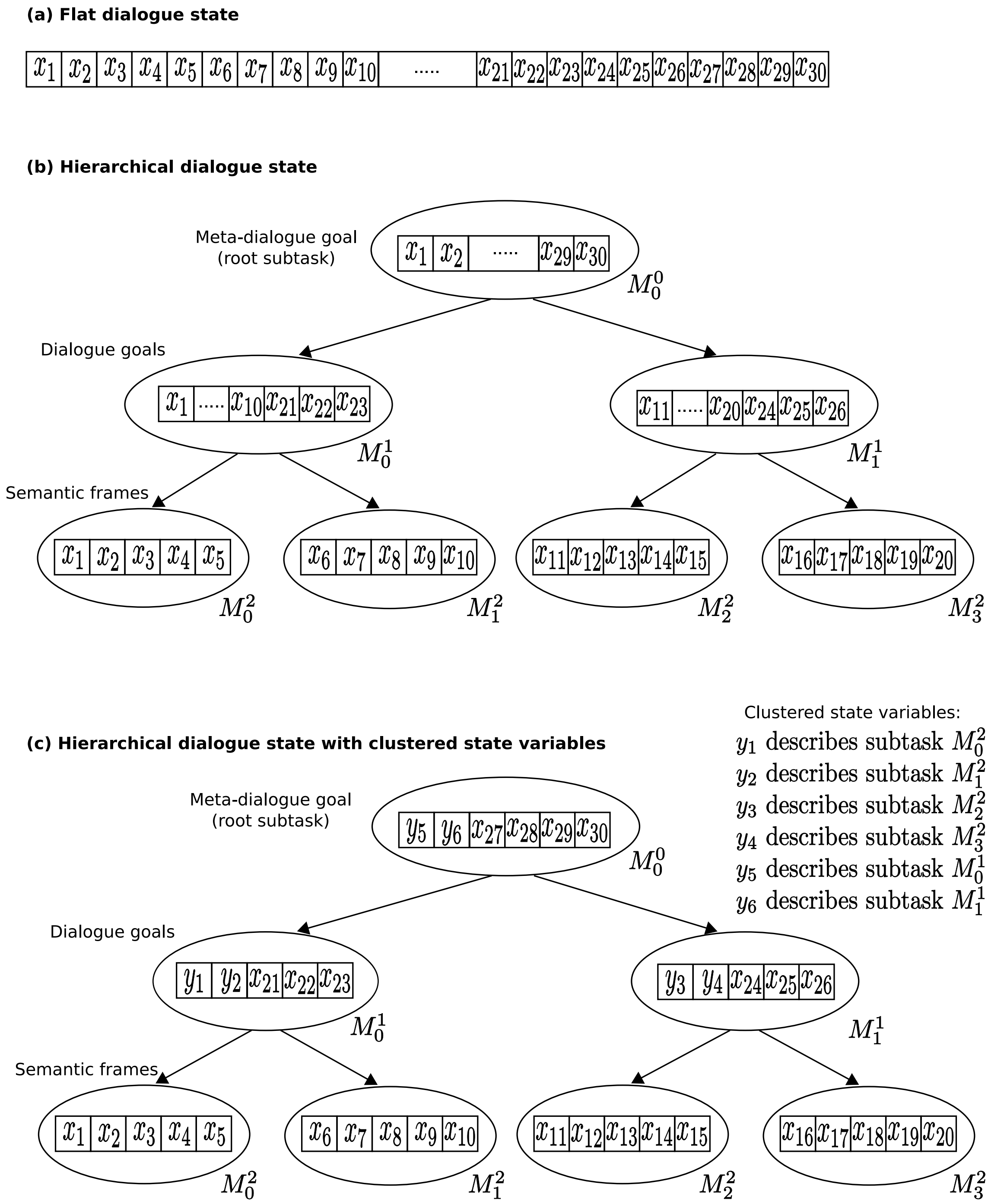


Figure 5.4: *Conceptual example of heuristic dialogue state abstraction showing: (a) a dialogue state with the full set of state variables, (b) a hierarchical dialogue state ignoring irrelevant state variables per subtask, and (c) a more compact representation of the hierarchical dialogue state based on clustered state variables describing the status of child dialogue subtasks.*

It can be noted that the subtasks at the bottom of the hierarchy use a smaller number of state variables for decision-making, and parent subtasks use a larger number of state variables because they have to take into account their children's knowledge to make decisions. However, the knowledge of the child subtasks can be represented more compactly in a parent subtask, which can be considered as knowledge at higher levels of granularity. Consequently, the subtask at the top of the hierarchy uses a compressed knowledge of the world by ignoring details only relevant for decision-making at lower levels in the hierarchy. For example: the meta-dialogue goal in Figure 5.4(c) ignores most of the information used for slot filling in the semantic frames.

5.2.3 Execution of dialogue subtasks

So far it has been said that a spoken dialogue manager can be defined by a hierarchy of dialogue subtasks $\mathcal{M} = \{M^i_j\}$, and that each subtask can apply state abstraction to compress the state space. The indexes i and j only identify a subtask in a unique way in the hierarchy, they do not specify the execution sequence of subtasks because that is learnt by the reinforcement learning agent. The execution of dialogue subtasks uses a stack and operates as follows: the dialogue starts with the root subtask M^0_0 in the stack; when a child subtask M^1_j is selected, it is pushed into the stack and control is transferred to the child subtask which is executed until reaching a terminal state – this may involve a recursive execution of other subtasks that may reach the bottom of the hierarchy; then the current subtask is popped off the stack and control is transferred back to the parent subtask at the next state $s' \in S^i_j$; and so on until the execution of the root subtask is completed, which empties the stack and terminates the dialogue.

5.2.4 Termination of dialogue subtasks

Typically, dialogue subtasks terminate when a goal has been reached; however, they may require a *temporal termination*. A spoken dialogue system might allow the user to go backwards or forwards in the conversation, i.e. move to different subtasks in the hierarchy. This requires a temporal termination of the current subtask, a move to another one, and a return to continue. The temporal termination may require to update the state variables of the current subtask and the clustered state variables at upper levels in the hierarchy, so that each subtask can choose actions accordingly. When a subtask terminates its execution, it is popped off the stack of dialogue subtasks. To allow a dialogue agent to abandon a sub-dialogue, the binary state variable 'END' can be added in a given subtask so that it can terminate in a deterministic way when $END = 1$. This allows early subtask termination in the required dialogue subtasks.

5.2.5 State transitions in SMDP-based dialogue optimization

Due to the fact that dialogue coherence is crucial for real-world spoken dialogue systems, two different kinds of states were employed in the SMDPs: (a) knowledge-rich states k_t and (b) knowledge-compact states s_t. Whilst the former include all possible information about the conversation, the latter include only a subset of it. Knowledge-rich states do not enumerate the vast combinations, they store only the current state of the world. These states hold attribute-values represented in an ontology-based structure. In contrast, knowledge-compact states – used to choose actions – enumerate a compact number of combinations. This implies non-deterministic state transitions in the SMDPs at the knowledge-rich level, which is due to stochastic user simulation and ASR error modelling (see chapter 4 for more details about the simulated dialogue environment). Figure 5.5 shows the dynamics in a dialogue-based SMDP. In addition, Figure 5.6 shows an illustrative example at runtime of knowledge-rich and knowledge-compact states for dialogue-based SMDPs.

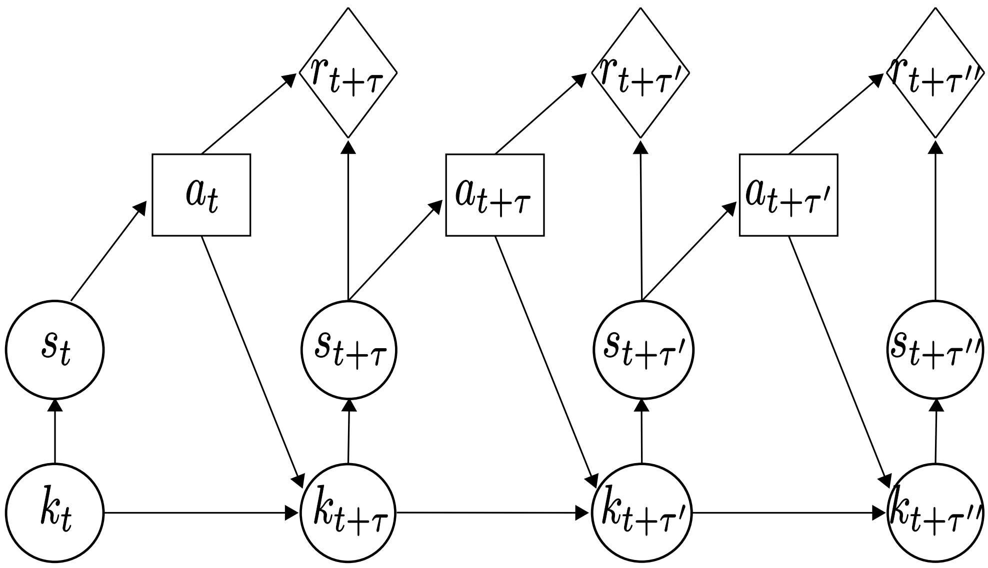


Figure 5.5: *An SMDP for spoken dialogue control. Notation: bottom circles represent knowledge-rich states, upper circles represent knowledge-compact states, rectangles represent actions, and diamonds represent rewards. The dynamics indicate that dialogue states s_t are observed from knowledge states k_t, and actions a_t can be either primitive (executed within the same SMDP) or composite (invoke a child SMDP).*

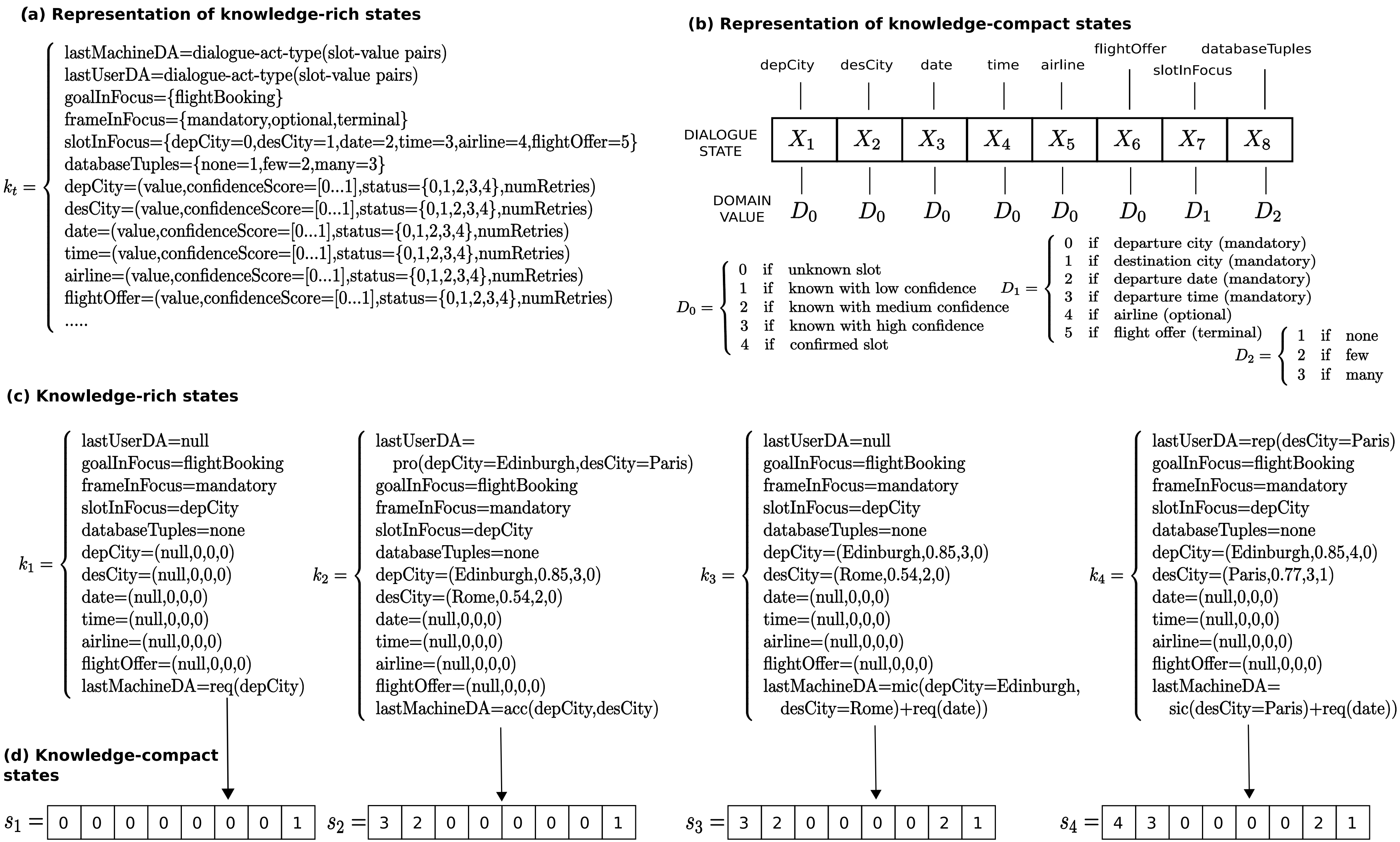


Figure 5.6: *Example in the flight booking domain of knowledge-rich states k_t and knowledge-compact states s_t for dialogue-based SMDPs – note that only the latter states are used for decision-making. Whilst (a) and (b) show the data structures for both states, (c) and (d) show those structures at runtime corresponding to the first four machine actions of the dialogue shown in page 35.*

5.3 Reinforcement learning for hierarchical SMDPs

The agent-environment interaction for dialogue control using hierarchical SMDPs is illustrated in Figure 5.7. Whilst the environment is modelled with a hierarchy of dialogue SMDPs, the learning agent takes action $a \in A_j^i$ in state $s \in S_j^i$ by using a hierarchy of policies executed with a top-down mechanism. Note that decision-making on each SMDP uses its corresponding policy, e.g. the behaviour in the root dialogue subtask M_0^0 follows policy $\pi_0^0(s)$. This section describes an algorithm that simultaneously learns a hierarchy of SMDP-based action-value functions $Q_j^{*i}(s,a)$. The approach described in this chapter differs from the MAXQ framework as follows: (1) the state abstraction per subtask is specified by the system developer, (2) it does not use pseudo-rewards, (3) the state transition function is based on knowledge-compact states derived from knowledge-rich states that store detailed information of the environment, and (4) the policy is executed only in a hierarchical way rather than (non) hierarchical.

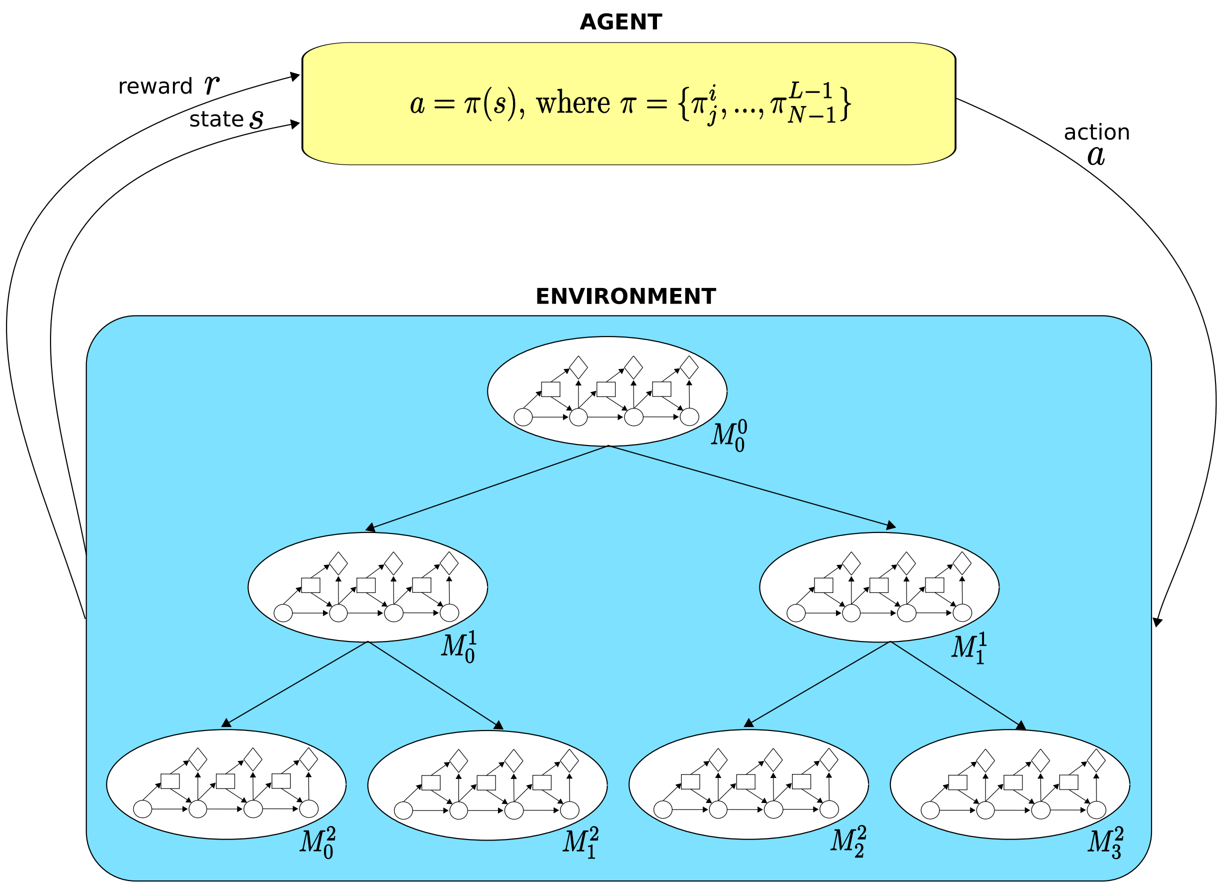


Figure 5.7: *Architecture of the agent-environment interaction for SMDP-based hierarchical reinforcement learning using a hierarchy of dialogue subtasks M_j^i. The subtasks are executed in a top-down hierarchical way using the well-known stack mechanism.*

Several methods have been investigated for learning a hierarchy of SMDPs such as Hierarchical Semi-Markov Q-Learning (HSMQ-Learning) (Dietterich, 2000b), where the action-value function Q^{*i}_j of equation 5.4 is approximated according to

$$Q^i_j(s_t, a_t) \leftarrow (1-\alpha) Q^i_j(s_t, a_t) + \alpha \left[r + \gamma^\tau \max_{a'} Q^i_j(s_{t+\tau}, a') \right]. \tag{5.6}$$

The summation over all τ time steps as appears in equation 5.4 is reflected here by using cumulative rewards $r = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^{\tau-t-1} r_{t+\tau}$ received for executing actions a_t, and by raising γ to the power τ. Algorithm 6 shows the procedural form of HSMQ-Learning adapted for handling knowledge-rich and knowledge-compact states. Briefly, this learning algorithm receives dialogue subtask M^i_j and knowledge base k used to initialize state s, performs similarly to Q-Learning for primitive actions, but for composite actions it invokes recursively with a child subtask. When the subtask is completed with τ time steps it returns a cumulative reward r at time $t+\tau$, and continues its execution until finding a terminal state for the root subtask M^0_0. This algorithm is iterated until convergence occurs to optimal context-independent policies (see page 50).

Algorithm 6 HSMQ-Learning with knowledge-rich and knowledge-compact states

function HSMQ(KnowledgeBase k, subtask M^i_j) **return** $totalReward$
  $s \leftarrow$ knowledge-compact state in S^i_j initialized from knowledge-rich state k
  $totalReward \leftarrow 0$, $discount \leftarrow 1$
  while s is not a terminal state **do**
    Choose action a from s using policy derived from Q^i_j (e.g. ϵ-greedy)
    Execute action a and update knowledge-rich state k
    if a is primitive **then**
      Observe one-step reward r
    else if a is composite **then**
      $r \leftarrow$ HSMQ(k, a), which invokes subtask a
        and returns the total reward received while a executed
    end if
    $totalReward \leftarrow totalReward + discount \times r$
    $discount \leftarrow discount \times \gamma$
    Observe resulting state s'
    $Q^i_j(s,a) \leftarrow (1-\alpha) Q^i_j(s,a) + \alpha \left[r + discount \times \max_{a'} Q^i_j(s', a') \right]$
    $s \leftarrow s'$
  end while
end function

5.4 Experimental setup

The aim of the experiments in this chapter was to investigate the potential application of the proposed approach to spoken dialogue systems with large state-action spaces. For such a purpose two rounds of experiments were performed. The first round of experiments compared flat versus hierarchical reinforcement learning when flat tabular learning is still feasible, and employed a 6-slot mixed-initiative dialogue system in the flight booking domain described in section 4.4.1. The second round of experiments were performed on a task where flat tabular reinforcement learning was no longer feasible, and employed a 26-slot mixed-initiative dialogue system in the travel planning domain described in section 4.4.2.

5.4.1 The flight booking case study

For flat reinforcement learning the state space representation has 8 non-binary state variables and 10 primitive actions. A description of the dialogue state variables is shown in page 34, and the action set is described in page 172. The reward function focused on efficient conversations (i.e. the shorter the dialogue the better), and is defined by the following rewards given to the agent for choosing action a when the environment makes a transition from state s to state s':

$$r(s,a,s') = \begin{cases} 0 & \text{for} \quad \text{successful (sub)dialogue} \\ \text{-10} & \text{for} \quad \text{presenting many/none items of information} \\ \text{-1} & \qquad\quad \text{otherwise.} \end{cases} \tag{5.7}$$

The execution of primitive actions applied the following consideration: illegal actions had no effect in the simulated dialogues and only wasted time, e.g. request an already filled slot, request an already confirmed slot, etc.

For hierarchical learning, the state-action space representation has 4 subtasks (one parent and three children); 11 non-binary state variables; 10 primitive actions and 3 composite actions. The latter correspond to the child subtasks. Figure 5.8 illustrates the subtask hierarchy and Table 5.1 shows the state variables and actions per subtask. It can be noted that the child subtasks are applying state abstraction by ignoring irrelevant variables. The root subtask is also applying state abstraction by using clustered state variables as follows: variable MAN represents the status of subtask M_0^1, variable OPT represents the status of subtask M_1^1, and variable TER represents the status of subtask M_2^1. In this way, the root subtask is using a much more compact version of the

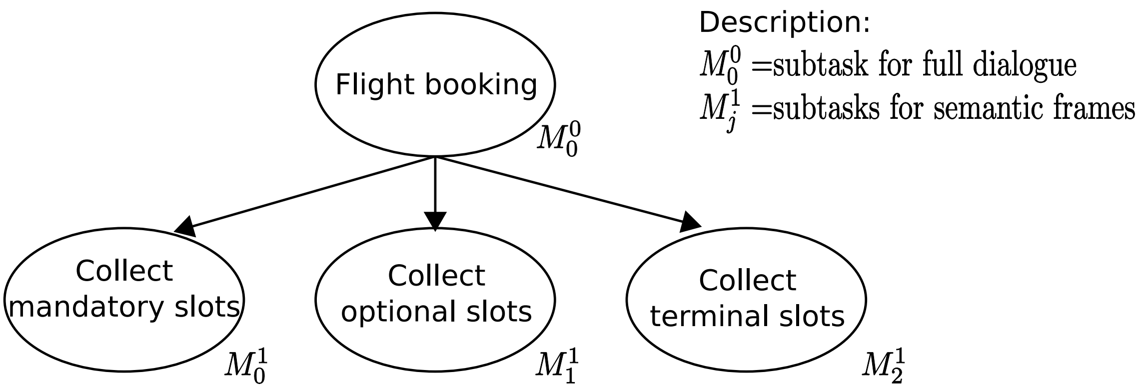


Figure 5.8: *A subtask hierarchy for the 6-slot flight booking spoken dialogue system, where each dialogue subtask is represented as a separate SMDP. The corresponding state variables and actions for each subtask M_j^i can be found in Table 5.1.*

Table 5.1: *State variables and actions of the subtask hierarchy in the flight booking spoken dialogue system (see Tables B.3 and B.4 for their corresponding description).*

#	Subtask	State Variables	Actions (*composite actions are* M_j^i)
01	M_0^0	MAN,OPT,TER,DBT	M_0^1,M_1^1,M_2^1,dbq+sta
02	M_0^1	SIF,C00,C01,C02,C03	req,apo+req,sic+req,mic+req,sec,mec,acc
03	M_1^1	C04	req,apo+req,sec
04	M_2^1	C05	pre+ofr,apo+ofr,sec

Note: the state variables {MAN, OPT, TER} represent clustered state variables from child subtasks and their domain values are as follows: {0=unfilled subtask, 1=filled subtasks, 2=confirmed subtask}.

dialogue state for decision-making. In addition, although a hierarchical reward function can be used for hierarchical dialogue optimization (i.e. a different reward function per subtask), these experiments used the same as in flat learning, used in each subtask.

The learning setup used Q-Learning for flat reinforcement learning (Watkins, 1989; Sutton and Barto, 1998) and HSMQ-Learning for hierarchical reinforcement learning (described in the previous section). The learning parameters used by the algorithms were the same for both learning approaches. The learning rate parameter α decays from 1 to 0 according to

$$\alpha = \frac{100}{(100+\tau)}, \tag{5.8}$$

where τ represents elapsed time-steps in the current subtask. Each subtask M_j^i had its own learning rate. The discount factor $\gamma = 1$ makes future rewards equally as valuable as immediate rewards, as in (Singh et al., 2002). The action selection strategy used ε-Greedy with $\varepsilon = 0.01$, and initial Q-values of 0. This choice of parameters satisfies the requirements for convergence to optimal (context-independent) policies.

5.4.2 The travel planning case study

This case study used a 26-slot mixed-initiative spoken dialogue system in the travel planning domain (see section 4.4.1 for a detailed description of this system), and is a larger-scale version of the previous case study. However, the experimental setup for flat tabular reinforcement learning is absent. This is due to the fact that using a single MDP for this task becomes impractical, the state space becomes too large to store ($\sim 10^{23}$ state-action pairs) and this makes the task intractable (due to memory limitations). In contrast, dialogue optimization for the travel planning spoken dialogue system becomes tractable within a hierarchical setting. This was possible by decomposing state variables and actions into a hierarchy of 21 subtasks including four levels of granularity. This hierarchy employed 43 non-binary state variables, 15 primitive actions and 20 composite actions. The latter correspond to the child subtasks. Figure 5.9 illustrates the subtask hierarchy and Table 5.2 presents the state variables and actions per dialogue subtask. The state abstraction used two sets of clustered state variables: {INI,MAN,OPT,TER} to describe the status of semantic frames, and {G00,G01,G02,G03,G04,G05} to describe the status of dialogue goals. The reward function also focused on efficient conversations (i.e. the shorter the dialogue the better), and is defined by the following rewards given to the agent for choosing action a when the environment makes a transition from state s to state s':

$$r(s,a,s') = \begin{cases} 0 & \text{for} \quad \text{successful (sub)dialogue} \\ \text{-10} & \text{for} \quad \text{an already collected subtask } M_j^i \\ \text{-10} & \text{for} \quad \text{collecting subtask } M_i^i \text{ before } M_{i-1}^i \\ \text{-10} & \text{for} \quad \text{presenting many/none items of information} \\ \text{-10} & \text{for} \quad \text{multiple greetings or closings} \\ \text{-1} & \quad\quad\;\; \text{otherwise} \end{cases} \tag{5.9}$$

The learning setup used the same parameters as in the previous case study.

The travel planning system allowed the user to go backwards in the dialogue and return to continue. The following is a sample scenario of early subtask termination. First, assume the user has filled and confirmed slots for a return flight (visiting subtasks M_0^3, M_1^3, M_3^3), so the current focus of the dialogue is in the terminal slot of return

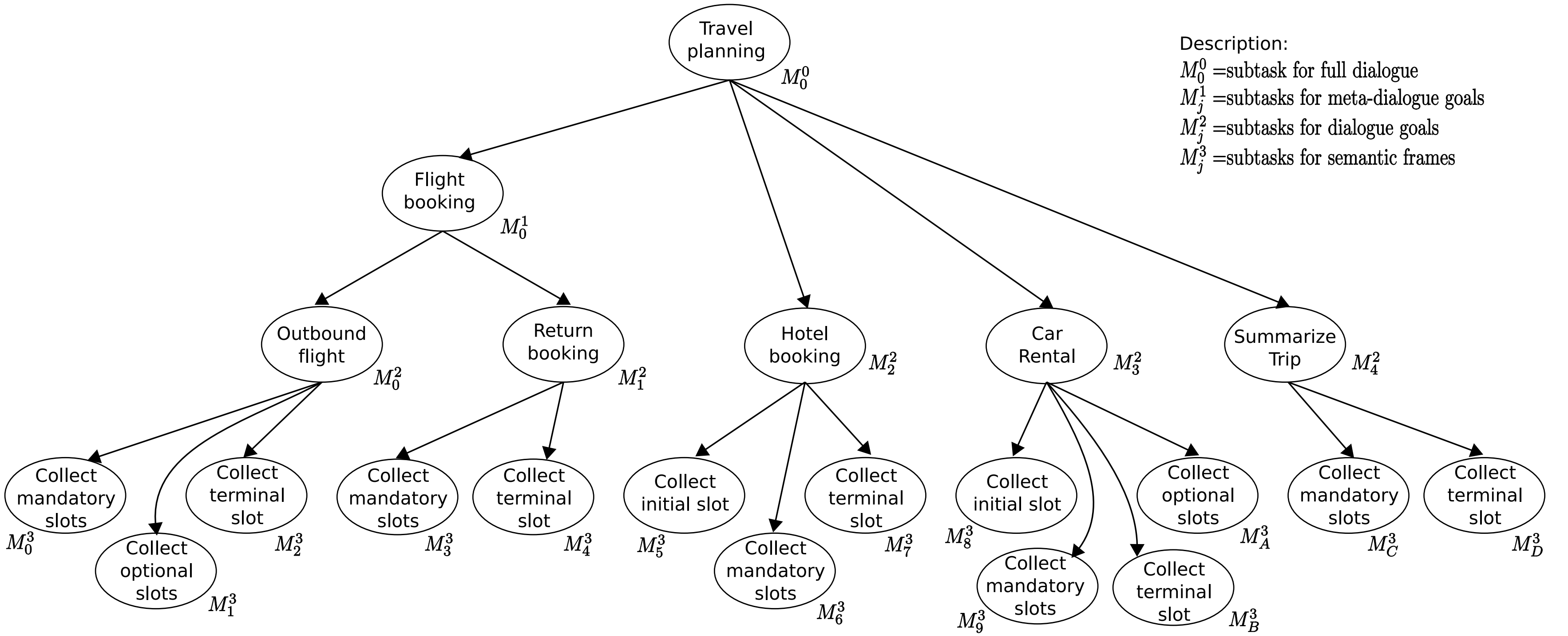


Figure 5.9: *A subtask hierarchy for the 26-slot travel planning spoken dialogue system, where each dialogue subtask is represented as a separate SMDP. The corresponding state variables and actions for each subtask M_j^i can be found in Table 5.2.*

Table 5.2: *State variables and actions of the subtask hierarchy in the travel planning spoken dialogue system (see Tables B.5, B.6, and B.7 for their corresponding description).*

Subtask	State Variables	Actions (*composite actions are* M_j^i)
M_0^0	GIF,SAL,G00,G03,G04,G05	M_0^1,M_2^2,M_3^2,M_4^2,gre,clo
M_0^1	GIF,G01,G02	M_0^2,M_1^2
M_0^2	DBT,END,MAN,OPT,TER	M_0^3,M_1^3,M_2^3,dbq+sta,rel
M_1^2	DBT,END,MAN,TER	M_3^3,M_4^3,dbq+sta,rel
M_2^2	DBT,END,INI,MAN,TER	M_5^3,M_6^3,M_7^3,dbq+sta,rel
M_3^2	DBT,END,INI,MAN, OPT,TER	M_8^3,M_9^3,M_A^3,M_B^3, dbq+sta,rel
M_4^2	DBT,END,MAN,TER	M_C^3,M_D^3,dbq+sta,rel
M_0^3	SIF,C00,C01,C02,C03,C04, C05	req,apo+req,sic+req,mic+req, sec,mec,acc
M_1^3	C6	req,apo+req,sec
M_2^3	ACK,END,PRE,C07	apo+ofr,sec,pre+ofr,ofr,ack
M_3^3	SIF,C15,C16	req,apo+req,sic+req,mic+req, sec,mec,acc
M_4^3	ACK,END,PRE,C17	apo+ofr,sec,pre+ofr,ofr,ack
M_5^3	C18	req,apo+req,sec
M_6^3	SIF,C19,C20,C21	req,apo+req,sic+req,mic+req, sec,mec,acc
M_7^3	ACK,END,PRE,C22	apo+ofr,sec,pre+ofr,ofr,ack
M_8^3	C23	req,apo+req,sec
M_9^3	SIF,C24,C25,C26,C27,C28	req,apo+req,sic+req,mic+req, sec,mec,acc
M_A^3	C29	req,apo+req,sec
M_B^3	ACK,END,PRE,C30	apo+ofr,sec,pre+ofr,ofr,ack
M_C^3	C31	req,apo+req,sec
M_D^3	ACK,END,PRE,C32	apo+ofr,sec,pre+ofr,ofr,ack

Notes: (1) the sets of state variables {INI, MAN, OPT, TER} and {G00, G01, G02, G03, G04, G05} represent clustered state variables from child subtasks and their domain values are as follows: {0=unfilled subtask, 1=filled subtasks, 2=confirmed subtask}. (2) the domain values of the state variable {END} are as follows: ={0=execution on the current subtask, 1=terminate the current subtask}.

flight (subtask M_4^3), but it turns out that the agent did not find flights with the provided information (subtask M_4^3 terminates), and then the agent invites the user to change some information. Second, the user reprovides information such as airline or departure date (go to subtask M_0^3 according to the stack of subtasks). Third, the agent searches flights again when it returns to subtask M_1^2, and offers the flight information in subtask M_4^3. Notice that when the user provides or reprovides information, the state variables at different subtasks in the hierarchy may require to be updated, so that each subtask can choose actions accordingly.

5.5 Experimental results

This section reports experimental results on dialogue strategy learning for the two case studies described in the previous section. Both case studies used the simulated conversational environment and baseline machine dialogue behaviour described in chapter 4.

5.5.1 The flight booking dialogue system

Experimental results show that the hierarchical state-action space obtained a dramatic reduction of 99.36% in comparison with a flat state-action space. Table 5.3 shows the number of state-actions for both flat (2.8 million) and hierarchical (17.8K) approaches.

Table 5.3: *Size of state-action spaces for the flight booking dialogue system.*

Approach	States	Actions	$\|S \times A\|$
Flat	281250	10	2812500
Hierarchical	2591	variable per subtask	17854

Figure 5.10 shows the learning curves of the dialogue policies, averaged over 10 training runs of 10^5 episodes (or dialogues). The three plots illustrate different distributions of ASR confidence levels. The first thing to notice is that hierarchical learning learnt faster than flat learning by roughly four orders of magnitude. The second thing to notice is that the hand-crafted strategy performed almost as well as the learnt policies for only one situation, but in general it was outperformed by the learnt policies. This illustrates the benefits of using dialogue optimization where more efficient conversations can be achieved by using (near) optimal dialogue strategies. The fact that the quality of the learnt policies are dependent on the simulation parameters suggests

that the simulation environment must reflect as much as possible the behaviour of the real environment, otherwise the learnt dialogue policies will no longer be optimal. The last thing to notice is that flat learning eventually performed slightly better than hierarchical learning. An evaluation on the last 10^4 episodes (dialogues) reports that flat learning achieved slightly more efficient conversations, on average 0.3 system turns fewer than hierarchical learning (significant at $p < 0.01$ for all confidence level distributions derived from t-tests). This is presumably because in the hierarchical setting the optional slot ('airline') cannot be confirmed together with the mandatory slots. Nevertheless, for practical purposes this loss in optimality may be well worth the gains in terms of scalability to larger decision-making problems.

5.5.2 The travel planning dialogue system

Experimental results show that the hierarchical state-action space also obtained a dramatic reduction of more than 99.99% in comparison with a flat state-action space. Table 5.4 shows the state-actions for both flat (10^{23}) and hierarchical ($800K$) approaches.

Table 5.4: *Size of state-action spaces for the travel planning dialogue system.*

Approach	States	Actions	Subtasks	$\|S \times A\|$
Flat	4.5×10^{22}	15	1	6.7×10^{23}
Hierarchical	117081	variable per subtask	21	803627

Figure 5.11 shows the learning curves of the dialogue policies, averaged over 10 training runs of 10^5 episodes (or dialogues). The three plots also illustrate different amounts of ASR confidence levels. In a similar way to the flight booking system, it can be observed that the hand-crafted strategy performed as well as the learnt policies only in the situation where $p(high) = 1/2$ (top plot of Figure 5.11), but in general it was outperformed by the hierarchical learnt dialogue policies. It can also be noted that the learnt behaviour required at least four orders of magnitude (i.e. more than 10000 dialogues) to outperform the hand-crafted behaviour. Because the learning speed of the given experimental setting is slow, other experimental settings or methods can be used to accelerate learning (this is addressed later in this section and in chapter 6).

A manual inspection of test dialogues showed that the learnt dialogue strategies generated coherent conversations. But the learnt policies sometimes exhibited dialogues with infinite loops, i.e. action a in state s yielded the next state $s' = s$ cyclically.

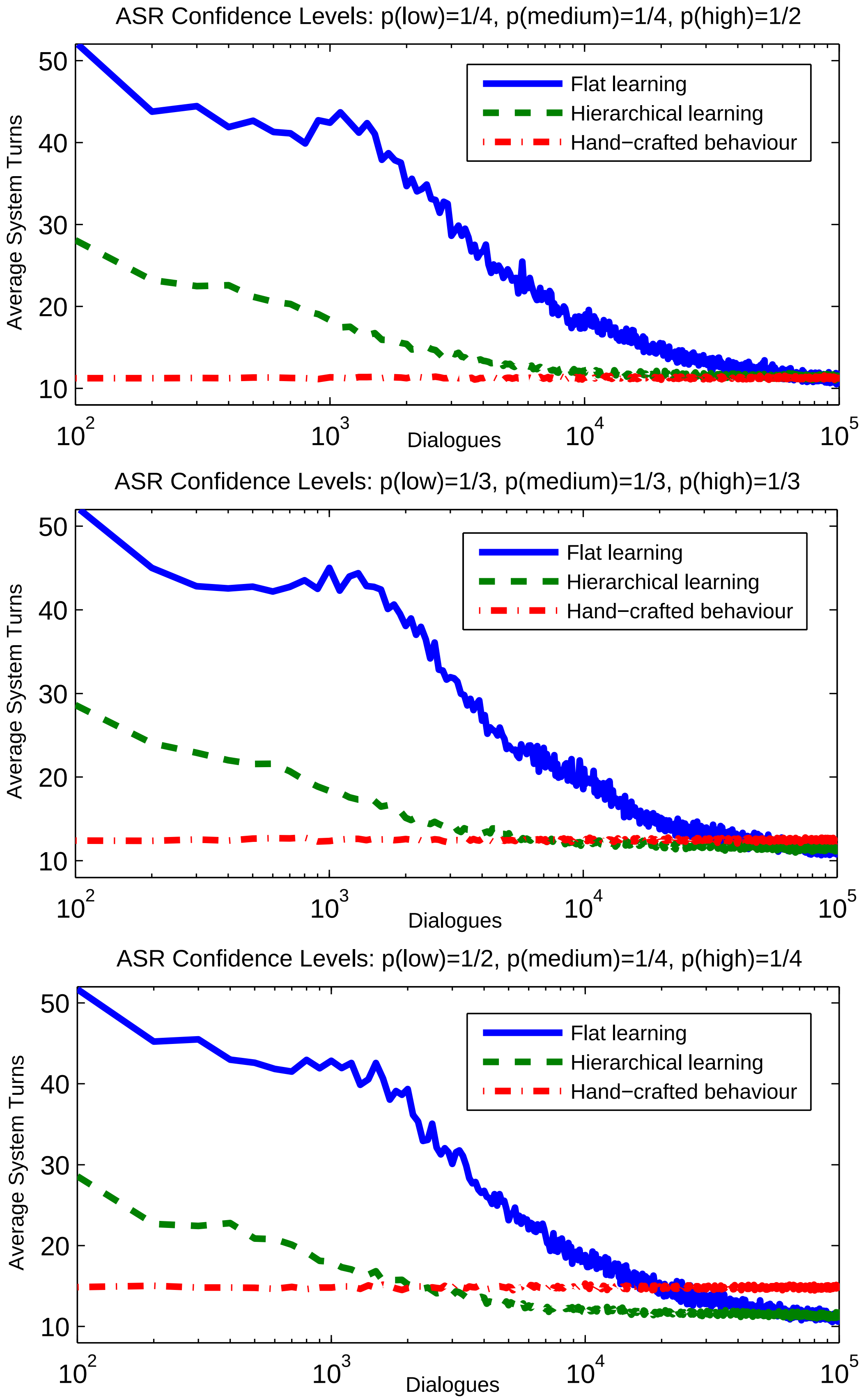


Figure 5.10: *Learning curves of dialogue policies in the 6-slot flight booking spoken dialogue system. The best learnt policy outperformed the hand-crafted behaviour by 0.2, 1.3, and 3.7 system turns on average in all cases (from top to bottom).*

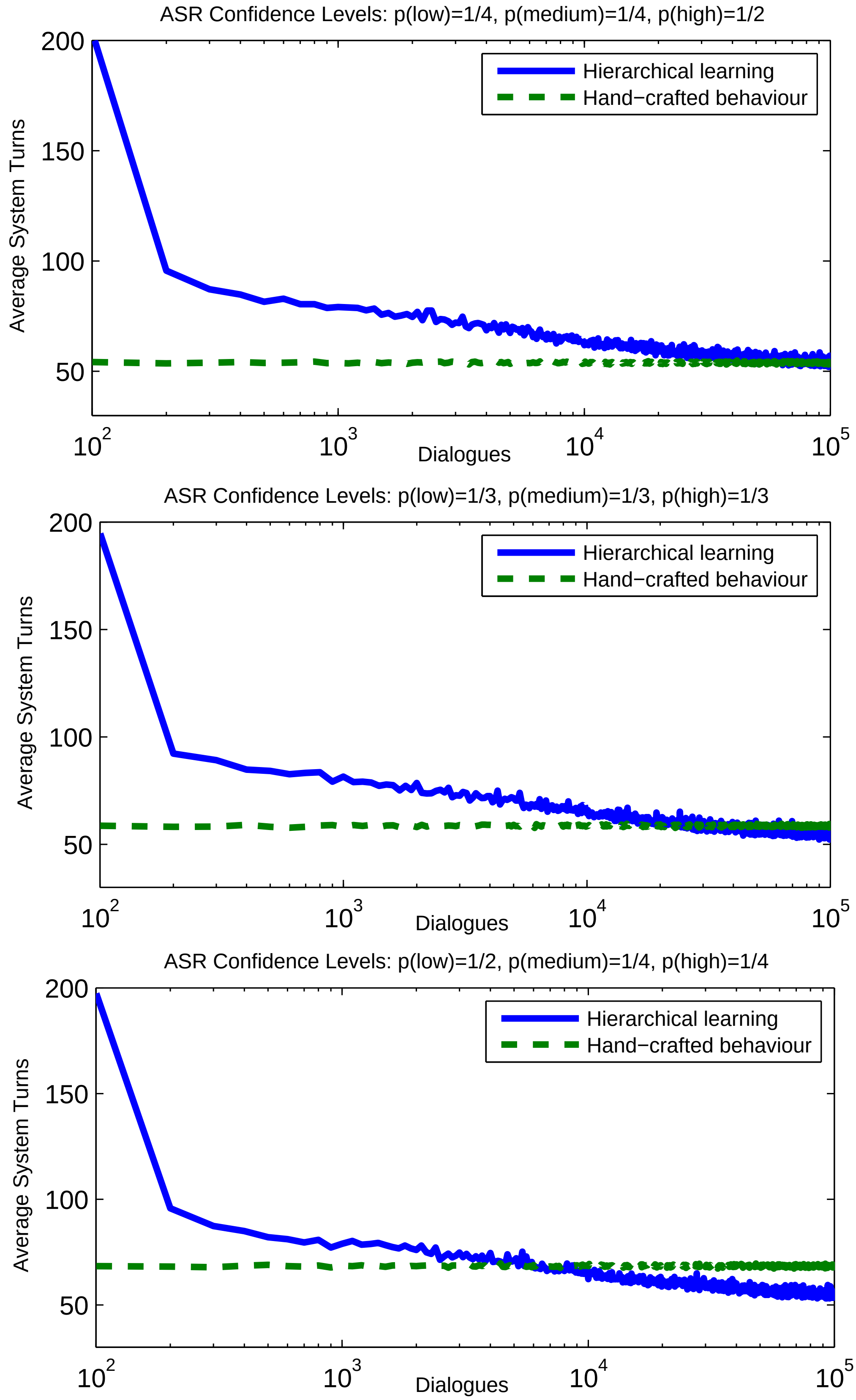


Figure 5.11: *Learning curves of dialogue policies in the 26-slot travel planning system using the reward function defined by equation 5.9. In the last* 10^4 *dialogues the hierarchical policy averaged* -0.2, 4.2, *and* 13.4 *fewer system turns than hand-crafted behaviour for the different distributions of confidence levels (from top to bottom).*

This was possible if the learnt policy inferred invalid actions which had no effect in the conversation and did not change the dialogue state. As a consequence, the learnt policy executed the same action in the same state in an infinite way. For example: the learnt policy performed apologies regardless of the confidence level and therefore apologized infinitely often. This phenomenon was not visible during learning due to the explorative behaviour, where policies eventually act randomly and can always reach a goal state. In contrast, testing only involves exploitation and made infinite loops visible. Our first attempt to avoid infinite dialogues consisted in extending the reward function with an additional negative reward assigned to state transitions with potential infinite loops. This reward function is expressed by modifying equation 5.9 by adding a condition which gives a reward of -10 when executing action a and remaining in the same state $s' = s$:

$$r(s,a,s') = \begin{cases} 0 & \text{for} \quad \text{successful (sub)dialogue} \\ -10 & \text{for} \quad \text{an already collected subtask } M^i_j \\ -10 & \text{for} \quad \text{collecting subtask } M^i_i \text{ before } M^i_{i-1} \\ -10 & \text{for} \quad \text{presenting many/none items of information} \\ -10 & \text{for} \quad \text{multiple greetings or closings} \\ -10 & \text{for} \quad \text{executing action } a \text{ and remaining in state } s' = s \\ -1 & \qquad\ \ \text{otherwise} \end{cases} \tag{5.10}$$

Figure 5.12 shows the learning curves of the dialogue policies using the reward function defined by equation 5.10. These learning curves were also averaged over 10 training runs of 10^5 episodes. It can be observed that hierarchical learnt dialogue behaviour outperformed hand-crafted behaviour for the three different distributions of confidence levels. It can also be noted that hierarchical learnt dialogue behaviour using the additional negative reward outperformed hand-crafted dialogue behaviour faster than the learning curves reported in Figure 5.11: (1) for optimistic confidence levels (top plot) the learnt policies significantly outperformed hand-crafted behaviour shortly after about 10000 dialogues, (2) for equal distributions the learnt policies outperformed hand-crafted behaviour by nearly 10000 dialogues, and (3) for pessimistic confidence levels (bottom plot) the learnt policies outperformed hand-crafted behaviour by nearly 1000 dialogues.

An evaluation of the last 10^4 dialogues reports that the hierarchical policy using the additional negative reward helped to reduce the problem of dialogues with infinite loops, but the learnt policies still exhibited infinite dialogues (a sample dialogue is shown in page 149). In addition, it was observed that the hierarchical policy using

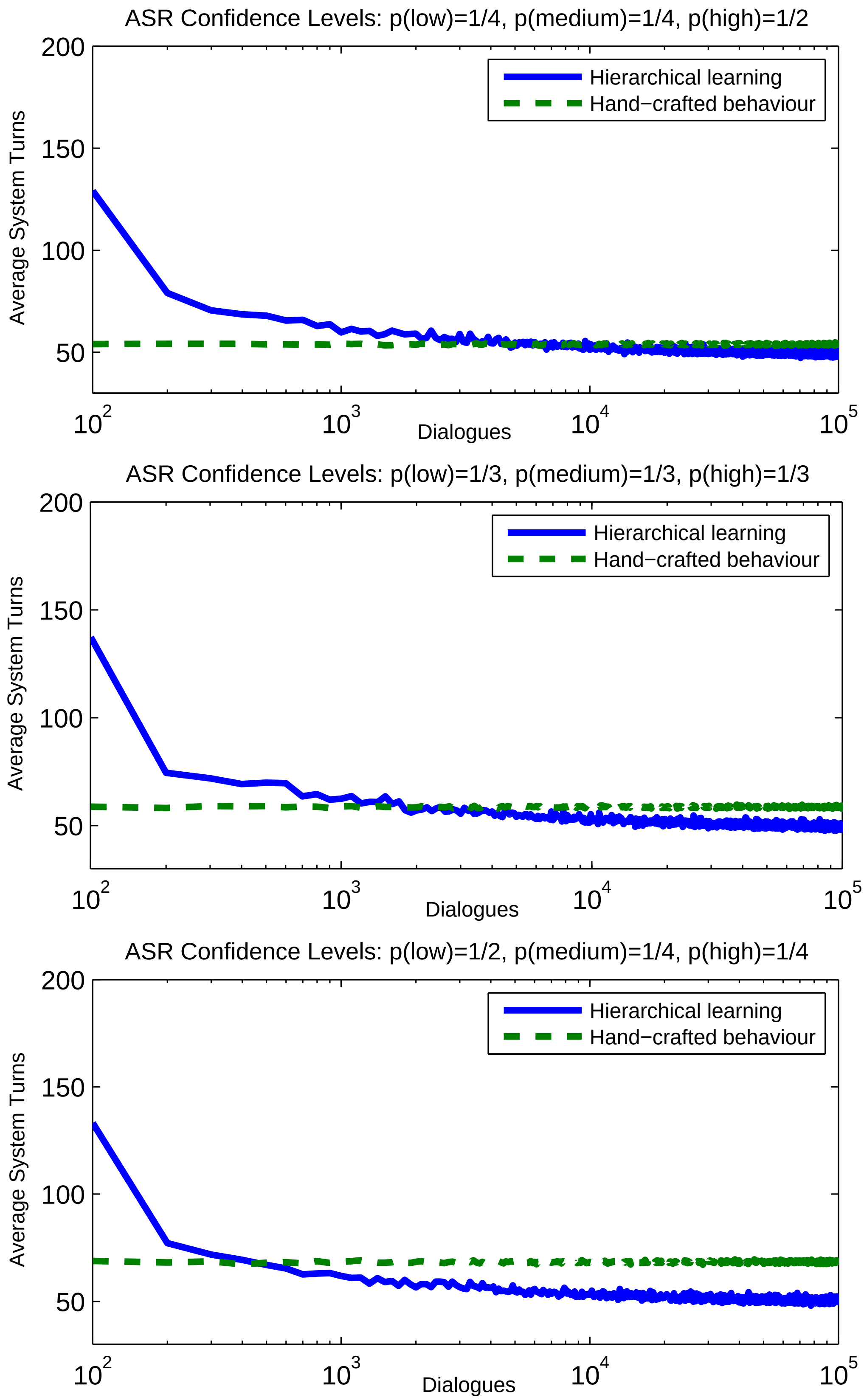


Figure 5.12: *Learning curves of dialogue policies in the 26-slot travel planning system using the reward function defined by equation 5.10. In the last* 10^4 *dialogues the hierarchical policy averaged* 4.9, 9.2, *and* 17.9 *fewer system turns than hand-crafted behaviour for the different distributions of confidence levels (from top to bottom).*

the additional negative reward generated more efficient conversations (see Table 5.5). This hierarchical policy outperformed the hand-crafted one by 4.9, 9.2, and 17.9 system turns for each distribution of confidence levels, respectively. This raises the following question: How well would such learnt policies perform in a realistic spoken dialogue environment? These results also suggest that other reward functions or mechanisms should be investigated for optimizing efficient and effective dialogue policies for fully-learnt dialogue behaviour. The next sub-section describes another alternative for avoiding dialogues with infinite loops. Nonetheless, all these results suggest that the proposed divide-and-conquer approach is a scalable way to address dialogue optimization with large state-action spaces.

Table 5.5: *Average system turns of policies in the last* 10^4 *training dialogues, where the third column used the reward function described by equation 5.9 and the fourth column used the reward function described by equation 5.10.*

Confidence Level Distribution (low, medium, high)	Hand-crafted Behaviour	Learnt Behaviour[1]	Learnt Behaviour[2]
Distribution1 $(1/4, 1/4, 1/2)$	53.9 ± 0.9	54.1 ± 3.42	49.0 ± 2.7
Distribution2 $(1/3, 1/3, 1/3)$	58.6 ± 1.1	54.4 ± 3.30	49.4 ± 2.7
Distribution3 $(1/2, 1/4, 1/4)$	68.4 ± 1.4	55.0 ± 3.49	50.5 ± 3.0

5.5.3 Analysis of learnt behaviour without infinite loops

Another way to address the problem of infinite dialogues is to employ stochastic action selection in states with infinite loops, and deterministic action selection in states without infinite loops, as suggested by (Ohta et al., 2003). Because learnt spoken dialogue policies must exhibit coherent behaviour, this thesis suggests to back off from learnt behaviour to hand-crafted behaviour when the execution of action a in state s yields next state $s' = s$, defined by

$$a = \begin{cases} \pi^*(s) & \text{if} \quad s <> s' \\ \pi^{det}(s) & \text{otherwise}, \end{cases} \tag{5.11}$$

where $\pi^*(s)$ is the learnt dialogue policy and $\pi^{det}(s)$ is a hand-crafted deterministic dialogue policy. Table 5.6 shows test results for hand-crafted and learnt behaviour, where the latter used equation 5.10 and behaved according to equation 5.11, averaged over 10 runs of 1000 dialogues.

Table 5.6: *Test results showing the average number of primitive actions per dialogue for hand-crafted and learnt behaviour, the latter used equation 5.10 and behaved according to equation 5.11. The average number of actions per dialogue (in bold) within each ASR confidence level distribution were compared with t-tests and showed statistical significance at* $p < 0.01$.

Conf. Levels	$(1/4, 1/4, 1/2)$		$(1/3, 1/3, 1/3)$		$(1/2, 1/4, 1/4)$	
Action	Hand-crafted	Learnt	Hand-crafted	Learnt	Hand-crafted	Learnt
acc	3.61	5.46	2.82	5.89	2.85	5.38
ack	4.02	4.03	4.02	4.03	4.02	4.03
apo+ofr	1.18	0.04	1.80	0.06	3.47	0.10
apo+req	4.53	1.00	7.23	1.32	14.23	1.63
clo	1.00	1.00	1.00	1.00	1.00	1.00
dbq+sta	4.47	4.43	4.47	4.44	4.46	4.43
gre	1.00	1.00	1.00	1.00	1.00	1.00
mec	5.84	1.94	6.87	2.02	7.63	2.02
mic+req	2.35	2.96	1.92	2.95	1.99	2.91
ofr	0.27	0.02	0.26	0.10	0.26	0.17
pre+ofr	4.02	4.25	4.02	4.19	4.02	4.10
rel	0.47	0.13	0.48	0.04	0.46	0.04
req	10.63	9.28	11.52	9.02	11.66	9.55
sec	10.09	9.00	10.55	8.68	10.46	9.30
sic+req	3.15	3.26	2.73	3.53	2.71	2.84
Sum	**56.63**	**47.80**	**60.68**	**48.28**	**70.24**	**48.50**

From the table above, it can be observed that the learnt dialogue behaviour outperformed the deterministic hand-coded one by 16%, 20% and 31% fewer system actions for each confidence level distribution, respectively. This reduction in the number of system actions can be explained as follows: the learnt behaviour differs from the hand-crafted one in the use of more acceptances (action ‘acc’), more multiple implicit confirmations (action ‘mic’), fewer apologies (actions ‘apo+req’ and ‘apo+ofr’), and fewer multiple explicit confirmations (action ‘mec’). In this way, the hierarchical reinforcement learning dialogue agents generated more efficient conversations.

5.6 Discussion

In this chapter the following issues are addressed for optimizing spoken dialogue behaviours of real world systems: (1) importance of hierarchical dialogue strategy learning, (2) uncertainty in spoken dialogue, (3) state representation, (4) reward function, (5) dialogues with infinite loops, and (6) learning from scratch.

First, the importance of hierarchical learning is to perform a more scalable global optimization for the full dialogue session. This form of learning is also important to optimize decision-making at different levels of granularity, where the design of the subtask sequence might not be easy to hand-craft. For instance, consider two subtasks that collect mandatory slots for a particular dialogue goal, where one of them collects slots with system-initiative and the other with mixed-initiative. Which dialogue subtask should be chosen at a given point in a conversation? This scenario requires learning at low and high levels in the hierarchy to result in a unified dialogue policy. For such a purpose, a hierarchical learning agent can employ a parent subtask in order to learn to decide when to invoke one or other of the subtasks. Moreover, the importance of hierarchical learning increases according to the complexity and size of state-action space of a given dialogue system. Experimental results showed that state abstraction helped to compress the size of the state space in a dramatic way. Compressing the state-action space per dialogue subtask produces faster learning, reduced computational demands, and opportunity to reuse sub-solutions[1]. All these benefits occur at the cost of sub-optimal solutions. For example, in the optimization of the flight booking system it was shown that hierarchical learning generated slightly longer dialogues than flat learning. This is still attractive for spoken dialogue systems assuming that exact optimality is not absolutely essential, as long as learnt behaviours show to be better than deterministic hand-crafted behaviours.

Second, a main criticism of this work is that the proposed optimization approach is not focusing on uncertainty in the dialogue state. However, this work can be enhanced with influence diagrams (Horvitz and Paek, 2000) or beliefs over slot values (Bohus and Rudnicky, 2005a, 2006). Alternatively, this work could be transferable to POMDPs (Roy et al., 2000; Pineau et al., 2001; Williams, 2006; Young et al., 2007; Thomson et al., 2008). In addition, a spoken dialogue manager can be viewed as two related agents: one in charge of knowledge updates, and the other in charge of choosing actions assuming accurate knowledge updates. This thesis focused on the latter.

[1] In this work subtask reuse was not explored and is left as future work.

Third, related work on dialogue strategy learning emphasizes that the state space must be kept as small as possible due to the large number of dialogues required to find optimal solutions. At the same time, the state representation must include enough information for making good decisions (Levin et al., 2000; Walker, 2000; Litman et al., 2000; Young, 2000). In this thesis heuristic state abstractions were used. Therefore, another enhancement to this work is to find the best state variables for each dialogue subtask in a more principled way using approaches such as feature selection (Paek and Chickering, 2005; Frampton and Lemon, 2006; Rieser and Lemon, 2006b) or automatic state abstraction (Dietterich, 2000a; Andre and Russell, 2002; Uther, 2002; Jong and Stone, 2005; Marthi et al., 2006; Jonsson, 2008).

Fourth, similar to the previous point is the issue of defining the reward function. There are many ways to specify a reward function, measuring dialogue efficiency (Young, 2000), user satisfaction (Walker, 2000), or a weighted combination of costs (Levin et al., 2000). This thesis focused on optimizing dialogue efficiency, which has been shown to be correlated with user satisfaction (Chu-Carroll and Nickerson, 2000; Litman and Pan, 2002).

Fifth, using the proposed dialogue optimization approach, it was found that learnt policies on full state-action spaces may include infinite loops. This phenomenon has not received attention in previous investigations because they mostly hand-craft the state and action spaces in order to find solutions on small search spaces. Although the problem of infinite loops can be avoided using stochastic action-selection as suggested by (Ohta et al., 2003), this issue should be taken into account when learning dialogue policies using the whole action set per state. This research proposed to back off from learnt behaviour to hand-crafted behaviour in order to guarantee coherent action-selection (see sub-section 5.5.3).

Finally, another criticism of the proposed approach is that it involves unnecessary learning. If reinforcement learning agents learn from scratch, then they will explore many invalid state-actions, resulting in slow learning. Previous work in dialogue optimization performs rule-based state-action space reduction before learning, and lacks a principled approach for learning dialogue behaviour only where necessary. The next chapter addresses this issue.

5.7 Conclusions

This chapter proposed learning multiple dialogue strategies using hierarchical reinforcement learning under the formalism of Semi-Markov decision processes, where a hierarchy of policies is learnt instead of a single one. Its application to simulated spoken dialogue systems was investigated in the flight booking and travel planning domains, and the proposed approach was compared with flat reinforcement learning. This approach has not been applied before to dialogue and the results are promising. Experimental results confirmed those reported by researchers in reinforcement learning – hierarchical learning finds cheaper and faster solutions than flat learning with near-optimal policies. The hierarchical search space of the 6-slot case study used only 0.64% of the size of the flat search space. Results showed that hierarchical learning converged four orders of magnitude faster than flat reinforcement learning with a small loss in optimality (on average 0.3 system turns). In addition, the hierarchical search space of the 26-slot case study used fewer than 0.01% of the size of the flat search space. Results also showed that the learnt policies outperformed a hand-crafted one under three different situations of ASR confidence levels. Finally, our experiments reported that the proposed approach may generate dialogue policies with infinite loops. To that end, this chapter proposed backing off from learnt behaviour to a deterministic one in dialogue states with potential infinite loops, generating finite and coherent dialogues. All these results provide evidence to support the claim that the proposed approach can be successfully applied to spoken dialogue systems with large state-action spaces.

Chapter 6

Hierarchical dialogue optimization: a prior-knowledge approach

This chapter extends the approach in the previous chapter with constrained hierarchical Semi-Markov Decision Processes (SMDPs). Section 6.2 proposes the idea of partially specified dialogue strategies for optimizing constrained spoken dialogue controllers. Section 6.3 proposes a reinforcement learning method to solve a hierarchy of SMDPs constrained with prior expert knowledge. Section 6.4 reports experimental results with two dialogue systems in the flight booking and travel planning domains. Section 6.5 explains how the proposed approach differs from similar approaches in the field. Section 6.6 discusses the strengths and weaknesses of the proposed dialogue optimization approach. Finally, the last section summarizes the chapter and draws conclusions.

6.1 Introduction

The standard Reinforcement Learning (RL) framework assumes learning ab initio, without any prior knowledge of the dialogue task, limiting the scalability of RL agents to complex and real-world problems. Additionally, the use of learning agents that perform trial-and-error exploration without any prior knowledge could even be harmful or inappropriate in real environments. This makes more relevant the **role of prior knowledge** in reinforcement learning agents, with the central aim of constraining the search space. This offers the following benefits among others: (a) finding solutions faster, (b) reducing computational demands, (c) incorporating expert knowledge, (d) transfering knowledge across problems, and (e) scaling to larger problems. This is possible by adding a mechanism for pruning away invalid state-actions in the learning environment. However, its drawback is that sub-optimal solutions may be obtained. Nonetheless, good policies can be learnt according to the constraints specified.

In Reinforcement Learning (RL) for spoken dialogue systems little attention has been devoted to the incorporation of prior knowledge into the RL agents, and therefore to proposing principled ways of reducing search spaces to manageable sizes. Moreover, the role of prior knowledge in dialogue optimization is not only to find cheaper and faster solutions, but also to incorporate constraints due to system requirements provided by system designers or customers (Paek, 2006; Paek and Pieraccini, 2008).

Previous work in the literature of RL for spoken dialogue systems employs ad hoc rules to reduce the state-action space (Levin et al., 2000; Walker, 2000; Singh et al., 2002; Schatzmann et al., 2005b). Previously I proposed a generic state-action reduction algorithm to optimize confirmation strategies with the aim of avoiding unnecessary learning (Cuayáhuitl et al., 2006a). However, it turned out to be difficult to extend for more complex and larger scale dialogue systems. This preliminary work suggested that search space reduction before learning has the undesirable effect of requiring re-learning for every minor update to the dialogue behaviour. Thus, finding methods that facilitate the incorporation of prior expert knowledge into RL dialogue agents, and that reduce the re-learning effect, is of importance for their practical application.

To date work in the literature of artificial intelligence and machine learning has proposed several methods for incorporating prior knowledge into learning agents. Nilsson (1994); Benson and Nilsson (1996) employ 'teleo-reactive' agents with initially designed and self-modifiable behaviour operating in dynamic and uncertain environments. Other prior work employs hierarchical deterministic and stochastic Finite State Machines (FSMs) – referred to as 'Hierarchical Abstract Machines (HAMs)' – in order to incorporate prior knowledge into RL agents (Parr and Russell, 1997; Andre and Russell, 2000). FSMs are relatively simple to design, and are attractive because they match the way in which the behaviour of dialogue systems is typically specified.

This chapter proposes an approach to equip RL dialogue agents with prior expert knowledge. For such a purpose the HAMs of Parr and Russell (1997) are used to merge hand-coded and learnt dialogue behaviours (also referred to as **partially specified dialogue strategies**). Then HAM-based reinforcement learning is combined with the approach described in chapter 5, resulting in 'constrained hierarchical Semi-Markov Decision Processes (SMDPs)', which employ a hierarchy of SMDPs incorporating constraints. Experimental results indicate that the proposed combined approach is a flexible and efficient way of optimizing the behaviour of large-scale dialogue systems.

6.2 Partially specified dialogue strategies

The idea of partially specified dialogue strategies for conversational agents serves two important purposes. Firstly, to give freedom to the system developer in what to specify manually and what to optimize; and secondly, to reduce search spaces due to the fact that they grow exponentially using the standard RL framework. This idea was inspired by the Hierarchical Abstract Machines (HAMs) of (Parr and Russell, 1997). In a HAM, whilst the obvious actions (i.e. one reasonable action per state) are specified with deterministic transitions, the non-obvious actions (i.e. several reasonable actions per state) are specified with stochastic transitions. The latter is the behaviour to be learnt by the reinforcement learning agent. This brings the best of both deterministic and purely-learnt approaches for dialogue strategy optimization (Cuayáhuitl et al., 2006b).

As discussed in chapter 5, the idea of hierarchical dialogue optimization consists of finding a spoken dialogue controller that takes the best hierarchical actions (primitive or composite) for each different situation in the conversation. This chapter refines that idea with **constrained hierarchical dialogue optimization**, where dialogue states employ a reduced set of actions specified through HAMs. For such a purpose, the following methodology is proposed.

(i) Design a Markov Decision Process (MDP) by choosing an appropriate representation of states, actions and reward function.

(ii) Decompose the MDP into a hierarchy of Semi-Markov decision processes (SMDPs).

(iii) Design a partially specified dialogue strategy using HAMs, where the obvious behaviours, if any, are specified deterministically and the less obvious ones are specified stochastically.

(iv) Generate an induced hierarchy of SMDPs, where the actions are given by the HAMs, resulting in a more compact search space.

(v) Learn a hierarchy of dialogue policies using a simulated environment. Alternatively, learning could be performed on real conversations if data suffices.

(vi) Finally, test the quality of the learnt dialogue strategy.

The methodology described here is a variant of the one proposed by (Litman et al., 2000; Singh et al., 2002), and the differences are twofold: (a) hierarchical instead of flat dialogue optimization, and (b) a principled approach to specify prior knowledge in order to optimize constrained spoken dialogue controllers.

6.2.1 Dialogue control using constrained hierarchical SMDPs

An important extension to the approach of the previous chapter is to constrain each hierarchical SMDP with some prior expert knowledge, aiming to combine dialogue behaviour specified by human designers and behaviour automatically inferred by reinforcement learning agents. To that end, this thesis suggests associating a Hierarchical Abstract Machine (HAM) denoted as H_j^i to SMDP M_j^i in order to specify some prior expert knowledge (see section 3.2.1 for an introduction to reinforcement learning with HAMs). In this way, dialogue control can be seen as executing two decision-making models in parallel: a HAM, and a hierarchy of SMDPs. Each HAM partially specifies the behaviour of its corresponding subtask, and therefore constrains the actions that a reinforcement learning agent can take in each state. Figure 6.1 shows this form of dialogue control in which both models share decision-making. For such a purpose, a cross product of models per subtask is used, referred to as *induced SMDP* $M_j'^i = H_j^i \circ M_j^i$, see section 3.2.1 for details about the cross product. Briefly, the cross product operates as follows: (1) the induced state space uses joint states $(s, \bar{s})$, where s is an *environment state* in SMDP M_j^i and $\bar{s}$ is a *choice state* in HAM H_j^i; (2) a HAM tells its corresponding SMDP the available actions at state s; (3) the transition functions of both models are executed in parallel; and (4) the SMDP's reward function rewards each chosen primitive action. In this joint model the HAMs make decisions in states with a single action, and the policies of the SMDPs make decisions in states with multiple actions.

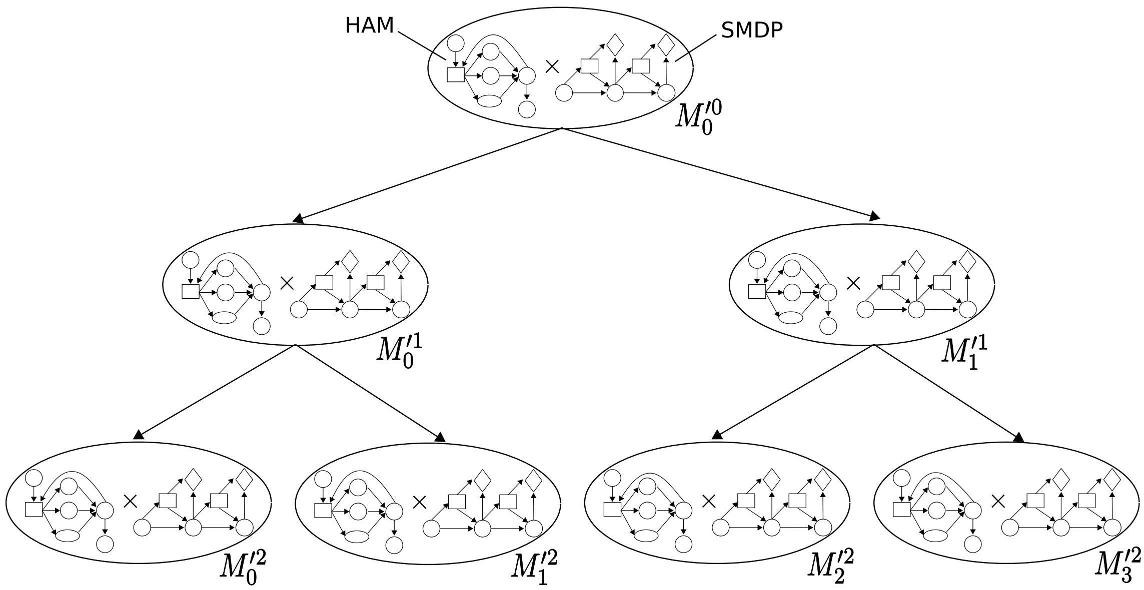


Figure 6.1: *Constrained hierarchical SMDPs are defined with induced SMDPs* $M_j'^i = H_j^i \circ M_j^i$*, where abstract machine* H_j^i *partially specifies the behaviour of subtask* M_j^i*.*

This form of dialogue control is based on SMDP state s and HAM choice state $\bar{s}$. Using a more compact notation for the joint dialogue state $w = (s, \bar{s})$ as in (Marthi et al., 2006), the Bellman equation for the action-value function of induced subtask $M_j'^i$ can be expressed as

$$Q_j^{*i}(w,a) = \sum_{w',\tau} P_j^i(w',\tau|w,a) \left[R_j^i(w',\tau|w,a) + \gamma^\tau \max_{a'} Q_j^{*i}(w',a') \right]. \qquad (6.1)$$

Optimal context-independent policies for the Q-value function above can be found by the learning algorithm described in section 6.3, and can be defined by

$$\pi_j^{*i}(w) = \arg\max_{a \in A_j^i} Q_j^{*i}(w,a). \qquad (6.2)$$

6.2.2 Decomposing a dialogue manager into subtasks

The decomposition of an MDP-based dialogue controller is carried out as in section 5.2.2. In a similar way, the prior expert knowledge can be decomposed into a Hierarchical Abstract Machine (HAM) $\mathcal{H} = \{H_j^i\}$. The cross product of HAM H_j^i and dialogue subtask M_j^i yields the induced subtask $M_j'^i = H_j^i \circ M_j^i$. But, if we want to reuse HAMs (e.g. a HAM for filling-confirming mandatory slots may be reused in all subtasks that collect mandatory slots) then they would have a more compact hierarchical structure that can be denoted as H_l^k, where $|H_l^k| \leq |H_j^i|$. Thus, the cross product of HAM H_l^k and subtask M_j^i yields the induced subtask $M_j'^i = H_l^k \circ M_j^i$ (see Figure 6.2).

6.2.3 Execution of dialogue subtasks

An induced dialogue subtask $M_j'^i$ is executed in a similar way as described in section 5.2.3. Briefly, when a subtask is invoked, it is pushed into a stack of subtasks, when it terminates its execution, it is popped off the stack, and the dialogue ends when the stack is empty. In addition, the parallel execution of HAMs and SMDPs operates as follows: when a subtask is invoked, the associated HAM takes control of the dialogue, control is transferred to the SMDP when the HAM is in a choice state; once the SMDP terminates the execution of the selected action it returns control to the HAM, and so on until termination of the root induced subtask.

6.2.4 Termination of dialogue subtasks

An induced subtask $M_j'^i$ terminates its execution as follows: (a) when the SMDP reaches a goal state, (b) when the SMDP makes an early termination (see section 5.2.4),

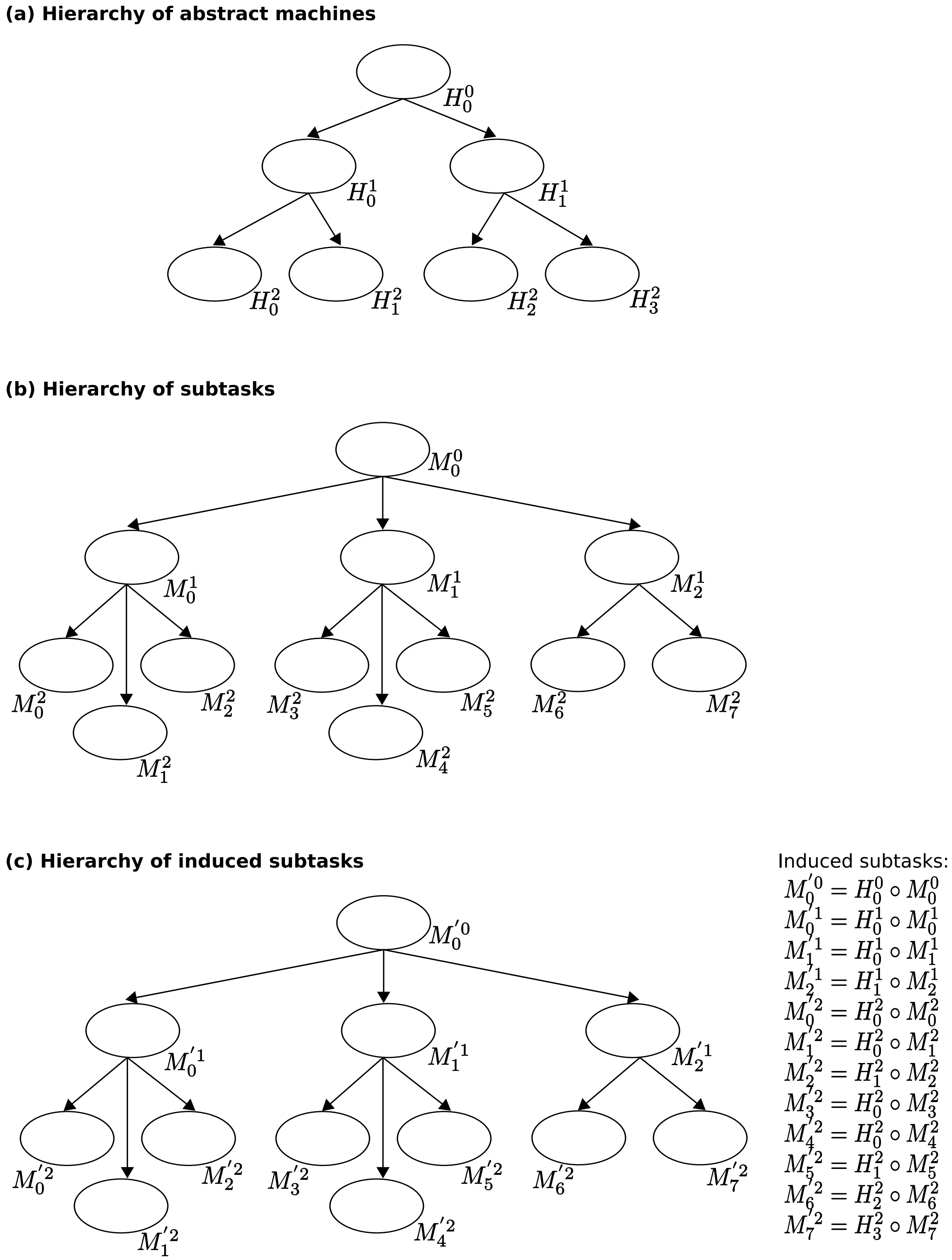


Figure 6.2: *Example of induced dialogue subtasks $M_j'^i = H_l^k \circ M_j^i$, where H_l^k is an abstract machine in $\mathcal{H}$ and M_j^i is a subtask in $\mathcal{M}$. Note that the hierarchy of abstract machines, Figure (a), and the hierarchy of dialogue subtasks, Figure (b), may be different because the abstract machines may be reused in the induced dialogue subtasks. The hierarchy of (induced) dialogue subtasks is specified by the system developer.*

or (c) when the HAM reaches a stop state. This suggests that HAMs should incorporate the termination conditions of their corresponding SMDPs. Thus, a HAM transitions to a stop state if and only if its corresponding SMDP has reached a terminal state.

6.2.5 State transitions in constrained hierarchical SMDPs

State transitions in constrained hierarchical SMDPs use three different types of states. Firstly, *knowledge-rich states* k_t include all possible information about the dialogue and do not enumerate the vast combinations, they only keep the current state of the world. Secondly, *knowledge-compact states* s_t include a subset of all information by enumerating a compact number of combinations. Thirdly, *machine states* $\bar{s}_n$ are states from a partially specified policy (HAM). The difference between this and the previous chapter is the inclusion of joint states $w_t = (s_t, \bar{s}_n)$, which are used by the reinforcement learning agent for decision-making. Figure 6.3 shows the dynamics of a constrained dialogue SMDP. In addition, Figure 6.4 shows an illustrative example at runtime of this form of dialogue control. Note that the indices of states $(s_t, \bar{s}_n)$ are different because knowledge-compact states s_t are only observed in machine choice states.

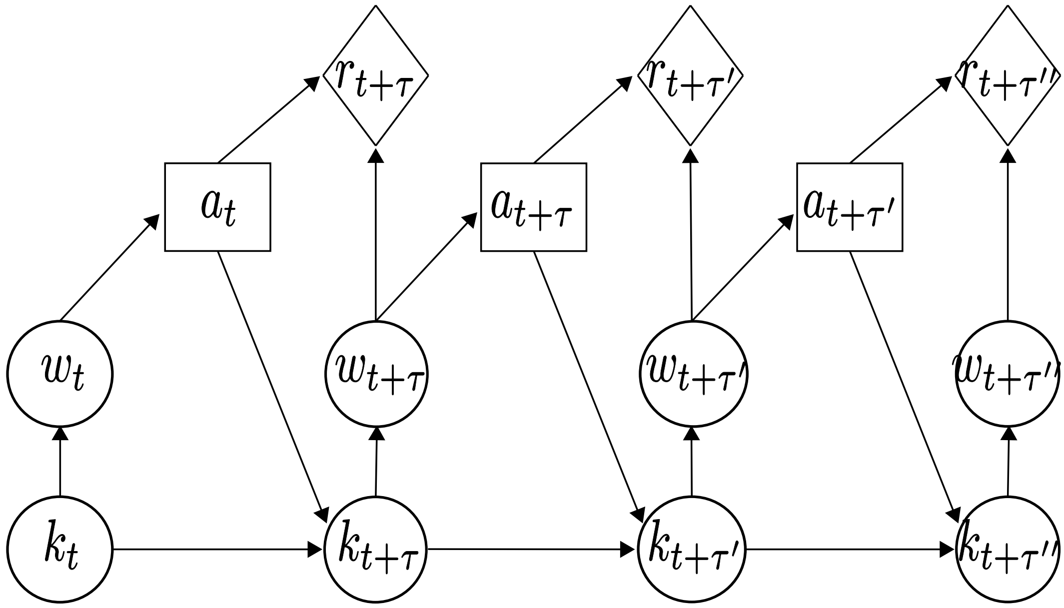


Figure 6.3: *A constrained SMDP for spoken dialogue control, where k_t represent knowledge-rich states, $w_t = (s_t, \bar{s}_n)$ represent joint states, rectangles represent actions (provided by a HAM), and diamonds represent rewards. Knowledge-compact states s_t, extracted from states k_t, are only observed in machine choice states $\bar{s}_n$, so that a restricted set of actions (primitive or composite) is to be available at dialogue state w_t.*

Machine states	Knowledge-rich states	Knowledge-compact states	Joint states	Actions available
$\bar{s}_1 = \text{start}$ $\bar{s}_2 = \text{choice2}$ $\bar{s}_3 = \text{req}$ $\bar{s}_4 = \text{null}$	$k_1 =$ lastUserDA=null goalInFocus=flightBooking frameInFocus=mandatory slotInFocus=depCity databaseTuples=none depCity=(null,0,0,0) desCity=(null,0,0,0) date=(null,0,0,0) time=(null,0,0,0) airline=(null,0,0,0) flightOffer=(null,0,0,0) lastMachineDA=req(depCity)	$s_1 = (0,0,0,0,0,0,0,1)$	$w_1 = (s_1, \bar{s}_2)$	{req}
$\bar{s}_5 = \text{start}$ $\bar{s}_6 = \text{choice4}$ $\bar{s}_7 = \text{acc}$ $\bar{s}_8 = \text{null}$	$k_2 =$ lastUserDA= pro(depCity=Edinburgh,desCity=Paris) goalInFocus=flightBooking frameInFocus=mandatory slotInFocus=depCity databaseTuples=none depCity=(Edinburgh,0.85,3,0) desCity=(Rome,0.54,2,0) date=(null,0,0,0) time=(null,0,0,0) airline=(null,0,0,0) flightOffer=(null,0,0,0) lastMachineDA=acc(depCity,desCity)	$s_2 = (3,2,0,0,0,0,0,1)$	$w_2 = (s_2, \bar{s}_6)$	{apo+req,acc,mec}
$\bar{s}_9 = \text{start}$ $\bar{s}_{10} = \text{choice2}$ $\bar{s}_{11} = \text{mic+req}$ $\bar{s}_{12} = \text{null}$	$k_3 =$ lastUserDA=null goalInFocus=flightBooking frameInFocus=mandatory slotInFocus=depCity databaseTuples=none depCity=(Edinburgh,0.85,3,0) desCity=(Rome,0.54,2,0) date=(null,0,0,0) time=(null,0,0,0) airline=(null,0,0,0) flightOffer=(null,0,0,0) lastMachineDA=mic(depCity=Edinburgh, desCity=Rome)+req(date))	$s_3 = (3,2,0,0,0,0,2,1)$	$w_3 = (s_3, \bar{s}_{10})$	{mic+req}
$\bar{s}_{13} = \text{start}$ $\bar{s}_{14} = \text{choice2}$ $\bar{s}_{15} = \text{sic+req}$ $\bar{s}_{16} = \text{null}$	$k_4 =$ lastUserDA=rep(desCity=Paris) goalInFocus=flightBooking frameInFocus=mandatory slotInFocus=depCity databaseTuples=none depCity=(Edinburgh,0.85,4,0) desCity=(Paris,0.77,3,1) date=(null,0,0,0) time=(null,0,0,0) airline=(null,0,0,0) flightOffer=(null,0,0,0) lastMachineDA= sic(desCity=Paris)+req(date))	$s_4 = (4,3,0,0,0,0,2,1)$	$w_4 = (s_4, \bar{s}_{14})$	{sic+req}
.....				

Figure 6.4: *Runtime example of HAM-based dialogue control using the abstract machine 'getMandatorySlots' from page 42. The first column shows a sequence of machine states corresponding to the first four primitive actions of the dialogue shown on page 44. The second and third columns show knowledge-rich states k_t and knowledge-compact states s_t that correspond to machine choice states $\bar{s}_n$. The fourth column shows joint states $w_t = (s_t, \bar{s}_n)$ used for decision-making. The last column shows the actions available in state w_t. The same example without machine states is shown in page 90.*

6.3 Reinforcement learning for constrained hierarchical SMDPs

The agent-environment interaction for constrained hierarchical dialogue control is shown in Figure 6.5. The environment is modelled with a hierarchy of induced SMDPs $M'^i_j = H^k_l \circ M^i_j$, where H^k_l is an abstract machine in the hierarchy of abstract machines $\mathcal{H}$, and M^i_j is an SMDP in the hierarchy of dialogue subtasks $\mathcal{M}$. The purpose of the abstract machine is to constrain the actions available per SMDP state. For such a purpose, the HAM-based reinforcement learning agent takes action $a \in A'^i_j$ in joint state $w = (s \in M^i_j, \bar{s} \in H^k_l)$ by using a hierarchy of policies π^i_j executed with a top-down mechanism. Note that joint states only include machine choice states, the remaining states are not taken into account by the reinforcement learning agent. For learning the hierarchy of policies we extend the HSMQ-Learning algorithm from the previous chapter with HAMQ-Learning (described in section 3.2.1). The algorithm described here differs from the HAM framework by using a hierarchy of SMDPs instead of a single SMDP. This algorithm simultaneously learns a hierarchy of action-value functions Q'^i_j, where equation 6.1 is approximated according to

$$Q'^i_j(w_t, a_t) \leftarrow (1-\alpha)Q'^i_j(w_t, a_t) + \alpha \left[r + \gamma^\tau \max_{a'} Q'^i_j(w_{t+\tau}, a') \right]. \tag{6.3}$$

In a similar way to the HSMQ-Learning algorithm described in the previous chapter, the summation over all τ time steps as appears in equation 6.1 is reflected here by using cumulative rewards $r = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{\tau-t-1} r_{t+\tau}$ received for executing actions a_t, and by raising γ to the power τ. The proposed learning algorithm for the hierarchy of induced SMDPs is called HAM+HSMQ-Learning. The procedural form of HAM+HSMQ-Learning is shown in algorithm 7. This reinforcement learning algorithm receives dialogue subtask M'^i_j and knowledge base k used to initialize state $w = (s, \bar{s})$. Then the abstract machine (corresponding to the current subtask) takes control of the interaction except in choice states, where the learning agent receives the control in order to choose actions; i.e. the abstract machine asks the learning agent how to act in choice states. This learning algorithm performs similarly to Q-Learning for primitive actions, but for composite actions it invokes recursively an induced subtask; when the induced subtask is completed with τ time steps it returns a cumulative reward r at time $t + \tau$, and so on until it finds a stop state $\bar{s}$ for the root induced dialogue subtask M'^0_0. The HAM+HSMQ-Learning algorithm is iterated until convergence occurs to optimal context-independent policies (see page 50).

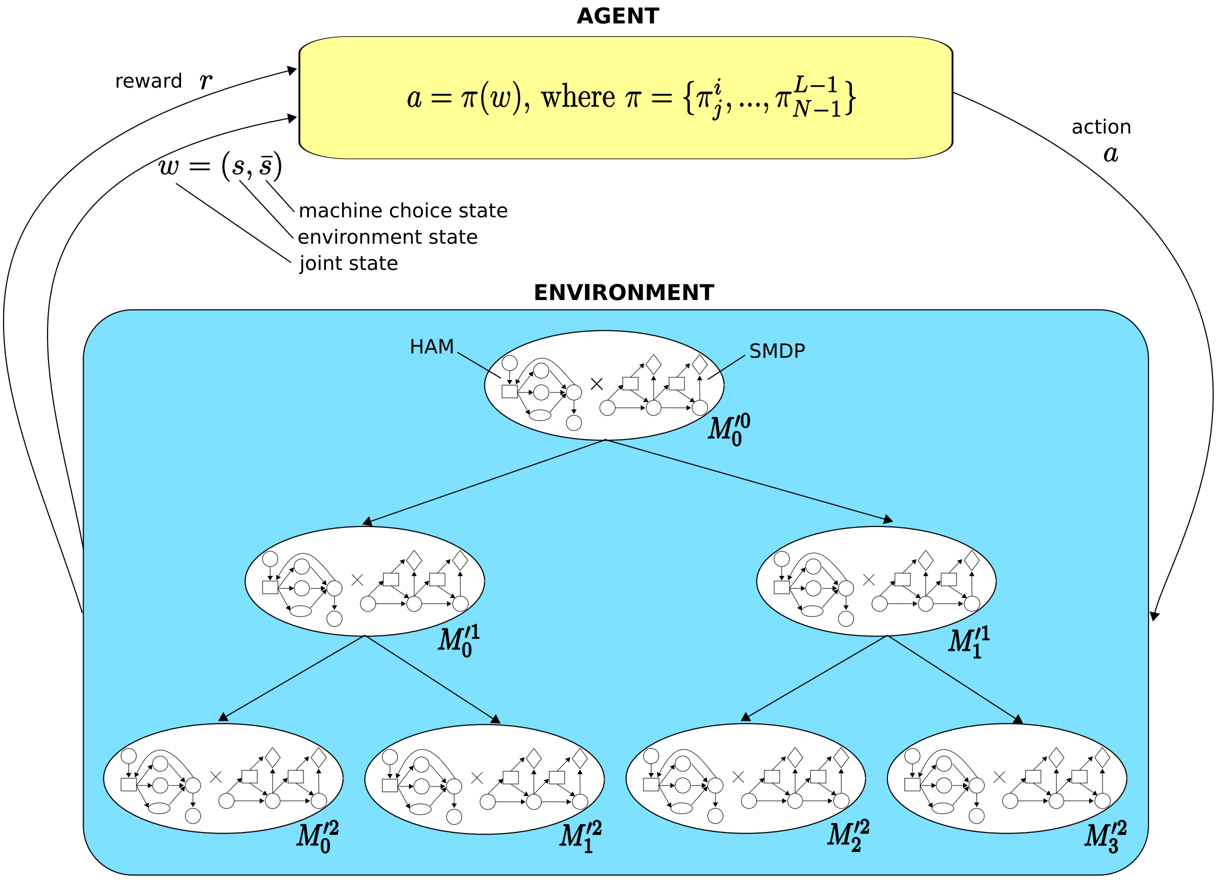


Figure 6.5: *Architecture of the agent-environment interaction for reinforcement learning using hierarchical induced SMDPs* $M'^i_j = H^k_l \circ M^i_j$. *The environment observes joint dialogue states* $w = (s, \bar{s})$, *where* s *is an environment state in SMDP* M^i_j *and* $\bar{s}$ *is a choice state in HAM* H^k_l. *The reinforcement learning agent uses a hierarchy of policies* π^i_j *for decision-making, where* i *denotes a level and* j *the model per level.*

Algorithm 7 The HAM+HSMQ-Learning algorithm

function HAM+HSMQ(KnowledgeBase k, subtask M'^{i}_{j}) **return** $totalReward$
  $s \leftarrow$ environment state in S^{i}_{j} initialized from knowledge-rich state k
  $\overline{s} \leftarrow$ start state of the abstract machine for subtask M'^{i}_{j}
  $w \leftarrow (s, \overline{s})$
  $totalReward \leftarrow 0$
  $discount \leftarrow 1$
  while $\overline{s}$ is not a stop state **do**
    if $\overline{s}$ is an action state **then**
      Execute action a (corresponding to $\overline{s}$) and update knowledge-rich state k
      Observe one-step reward r
    else if $\overline{s}$ is a call state **then**
      $r \leftarrow$ HAM+HSMQ(k, a), which invokes subtask a (corresponding to $\overline{s}$) and returns the total reward received whilst a executed
    else if $\overline{s}$ is a choice state **then**
      Chose action a from w using policy derived from Q^{i}_{j} (e.g. ε-greedy)
      $\overline{s} \leftarrow a$
      continue
    else
      Observe next machine state $\overline{s'}$ (e.g. a choice, null or stop state)
      $\overline{s} \leftarrow \overline{s'}$
      continue
    end if
    $totalReward \leftarrow totalReward + discount \times r$
    $discount \leftarrow discount \times \gamma$
    Observe resulting joint state $w' \leftarrow (s', \overline{s'})$
    $Q^{i}_{j}(w,a) \leftarrow (1-\alpha)Q^{i}_{j}(w,a) + \alpha \left[r + discount \times \max_{a'} Q^{i}_{j}(w',a') \right]$
    $\overline{s} \leftarrow \overline{s'}$
    $w \leftarrow w'$
  end while
end function

6.4 Experiments and results

The experiments reported here aimed to investigate dialogue systems that learn to behave from scratch against systems that learn to behave incorporating prior expert knowledge. As in chapter 5, the flight booking and travel planning systems used the simulated environment and baseline machine behaviour described in chapter 4.

6.4.1 Experimental setup

The experimental setup – in terms of state representations, actions, rewards and learning setup – was similar to that in section 5.4. The difference here is that the dialogue subtasks M^i_j were extended with Hierarchical Abstract Machines (HAMs) H^i_j, where their cross product yields the *induced subtasks* $M'^i_j = H^i_j \circ M^i_j$. The learnt policies used the algorithm HAM+HSMQ-Learning described in the previous section. The hierarchy of induced subtasks for the 6-slot flight booking system is shown in Figure 6.6, and used the same abstract machines described in chapter 3 (page 42). The hierarchy of induced subtasks for the 26-slot travel planning system is shown in Figure 6.7, and used the abstract machines described in Figures 6.8 and 6.9. These HAMs control the machine's dialogue behaviour in deterministic state transitions, but in stochastic state transitions the hierarchical reinforcement learning agents optimized decision-making. Note that whilst the flight booking system is not reusing abstract machines, the travel planning system is reusing abstract machines in several induced subtasks.

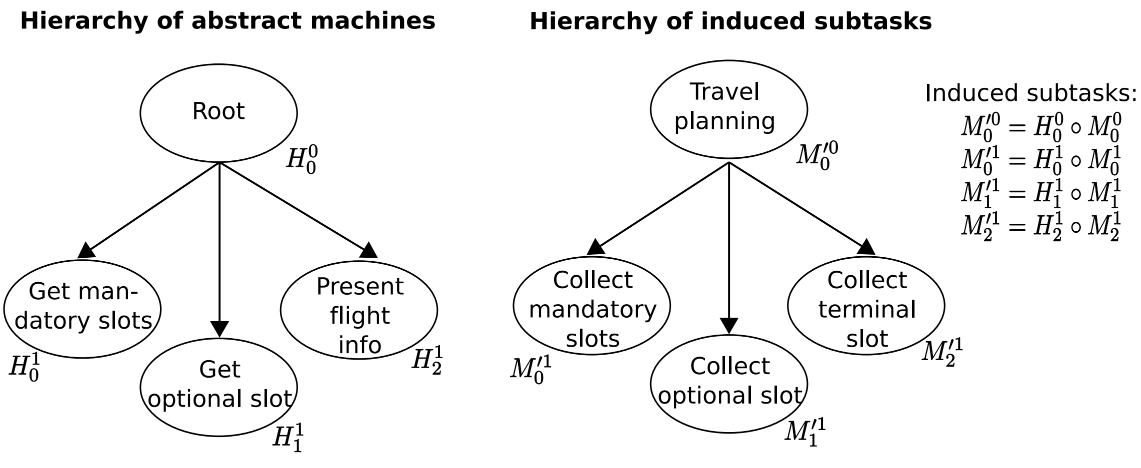


Figure 6.6: *A hierarchy of induced subtasks for the 6-slot flight booking spoken dialogue system. The abstract machines are specified in page 42 of chapter 3 and the state variables for each dialogue subtask M^i_j are specified in Table 5.1.*

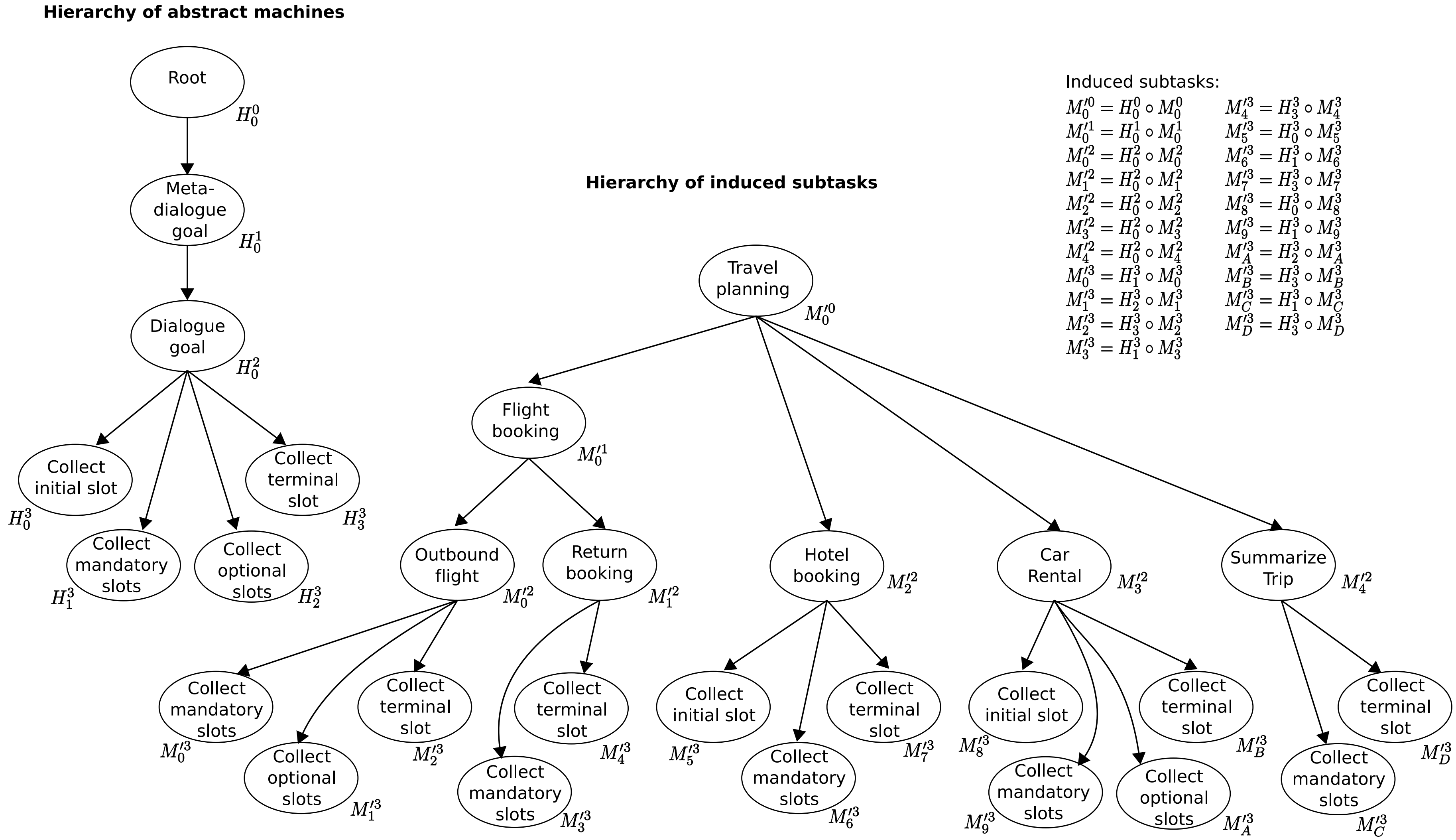


Figure 6.7: *A hierarchy of induced subtasks for the 26-slot travel planning spoken dialogue system. The abstract machines (denoted as H_l^k) are specified in Figures 6.8 and 6.9, and the state variables for each dialogue subtask M_j^i are specified in Table 5.2.*

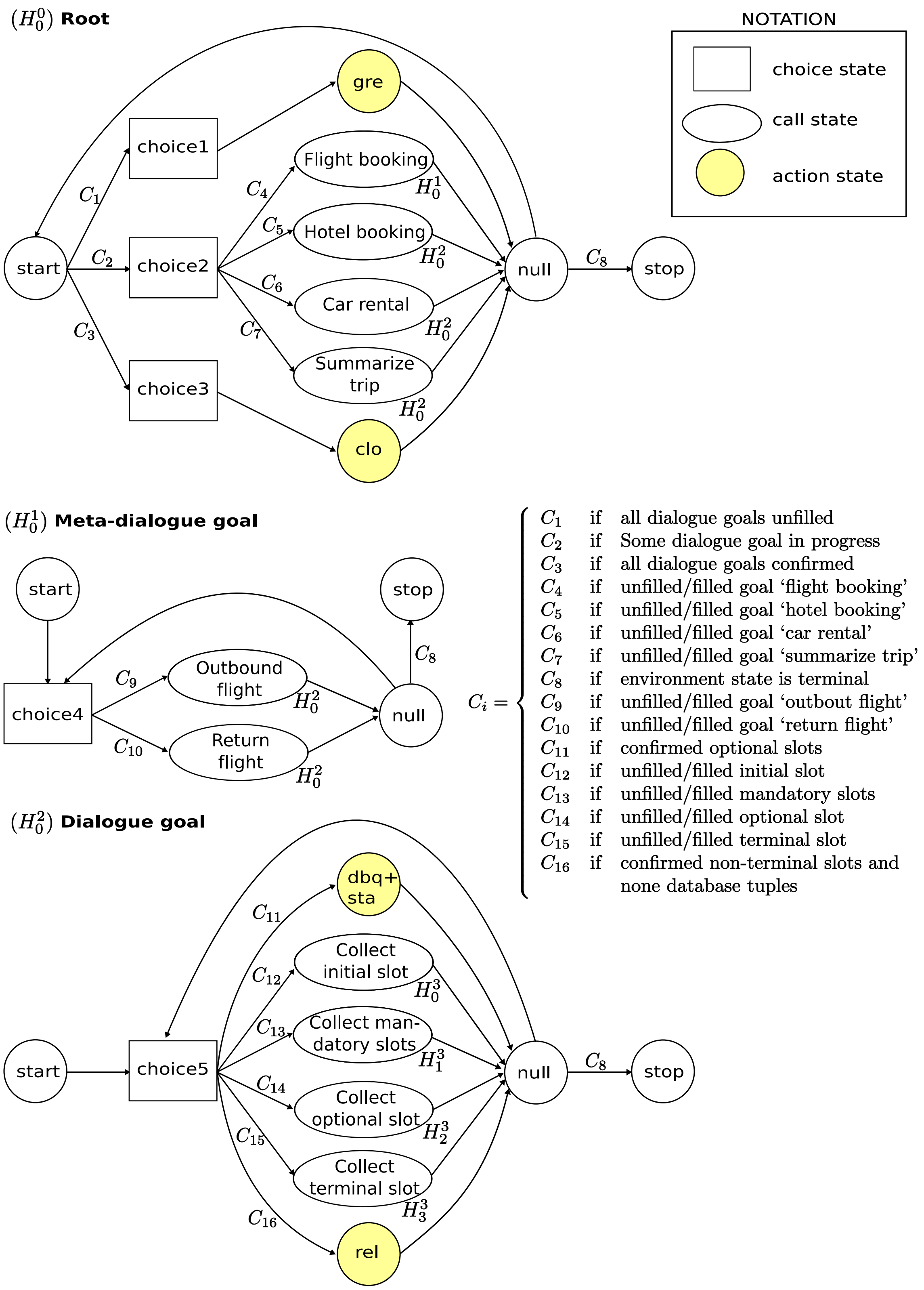


Figure 6.8: *Abstract machines for the travel planning spoken dialogue system (Part 1), where state transitions can be stochastic or based on deterministic constraints C_i.*

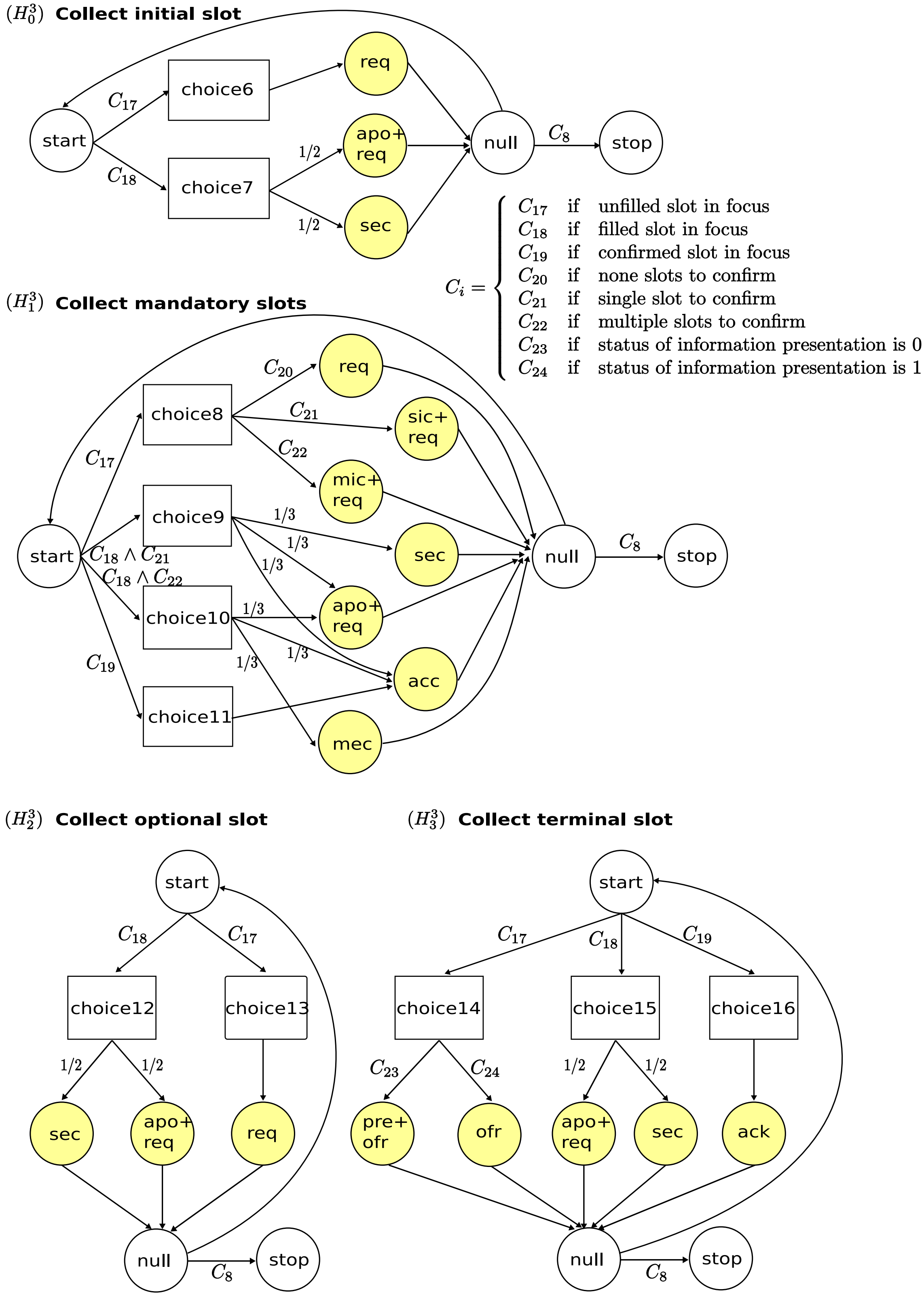


Figure 6.9: *Abstract machines for the travel planning spoken dialogue system (Part 2), where state transitions can be stochastic or based on deterministic constraints* C_i.

6.4.2 Experimental results: flight booking case study

Experimental results show that the hierarchical state-action space with HAM obtained a dramatic reduction of 99.80% in comparison with a flat state-action space. This represents an additional relative reduction of 67.28% to the hierarchical state-action space without HAM. Table 6.1 shows the number of state-actions for both flat (2.8 million) and hierarchical (17.8K and 5.8K) approaches. It can be observed that the state-action space reduction by the divide-and-conquer approach is much more significant than the prior knowledge approach. But the additional benefit of the latter approach is to perform learning on constrained dialogue behaviour.

Table 6.1: *Size of state-action spaces for the flight booking dialogue system.*

Approach	States	Actions	Subtasks	$\|S \times A\|$
Flat	281250	10	1	2812500
Hierarchical without HAM	2591	variable per subtask	4	17854
Hierarchical with HAM	2591	HAM-based	4	5841

Figure 6.10 shows the learning curves of the dialogue policies for different ASR confidence distributions, averaged over 10 training runs of 10^5 dialogues. The first thing to notice is that hierarchical learning with HAM learns faster than hierarchical learning without HAM, roughly by four orders of magnitude. The second thing to notice is that the HAM-based policy[1] required very little learning compared with learning from scratch. This can be explained by the fact that whilst the policy without HAM is exploring incoherent behaviour (by using the whole action set), the HAM-based policy is exploring more coherent behaviour. This is why the learning curve is flattened, but it gradually finds more efficient behaviour. The last thing to notice is that hand-crafted machine behaviour performed almost as well as the HAM-based policy for only one situation of confidence levels (top plot), but in general it is outperformed.

These plots report that the learnt dialogue policies outperformed the hand-crafted behaviour by 0.1, 1.2, and 3.4 system turns for the different distributions of confidence levels, respectively (from top to bottom). These gains in dialogue efficiency highlight the importance of validating these results with real conversations.

[1]The HAM-based policies used the following settings only in the first one hundred dialogues: (1) frozen learning, and (2) Q-values initialized to '1' for state-action pairs that matched the hand-crafted behaviour. This setting was employed to observe the policy's performance before learning.

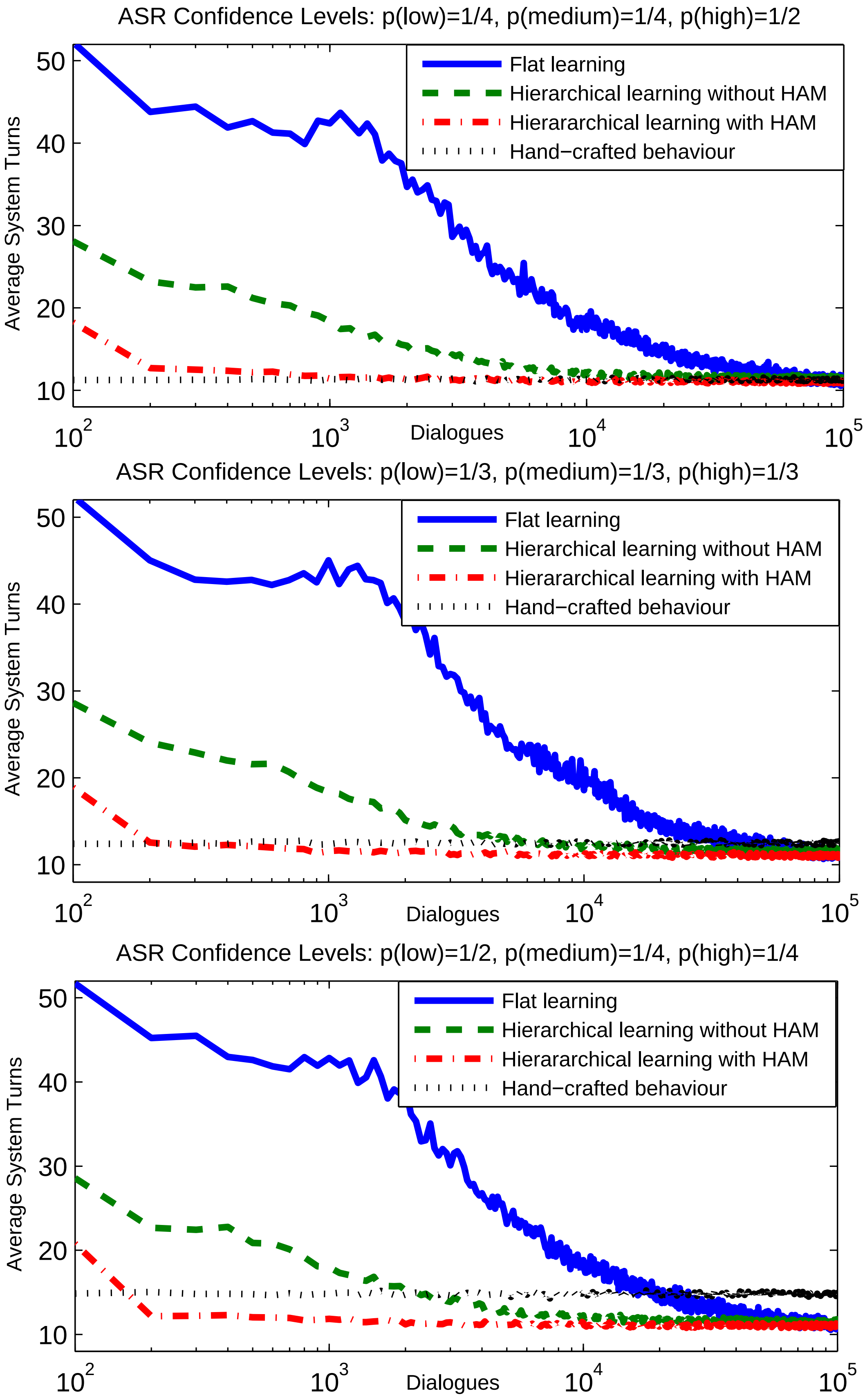


Figure 6.10: *Learning curves of dialogue policies using flat and hierarchical reinforcement learning (with and without prior knowledge) in the flight booking system.*

6.4.3 Experimental results: travel planning case study

Experimental results show that the hierarchical state-action space with HAM obtained a dramatic reduction of more than 99.99% in comparison to a flat state-action space. This represents an additional relative reduction of 69.37% compared to the hierarchical state-action space without HAM. Table 6.2 shows the number of state-actions for both flat and hierarchical approaches. But, how can good dialogue policies be found by throwing away more than 99.99% of state-actions?

Table 6.2: *Size of state-action spaces for the travel planning dialogue system.*

Approach	States	Actions	Subtasks	$\|S \times A\|$
Flat	4.5×10^{22}	15	1	6.7×10^{23}
Hierarchical without HAM	117081	variable per subtask	21	803627
Hierarchical with HAM	116457	variable per subtask	21	246171

Figure 6.11 shows the learning curves of hierarchical dialogue policies for different amounts of ASR confidence levels, also averaged over 10 training runs of 10^5 dialogues. Results confirm the arguments made in the previous case study. First, hierarchical learning with HAM found faster solutions than hierarchical learning without HAM. Whilst the former form of learning required less than 1000 dialogues to outperform hand-crafted behaviour, the later required at least 10000 dialogues to outperform hand-crafted behaviour. Second, HAM-based behaviour[2] required very little learning compared with learning from scratch. Third, HAM-based behaviour outperformed hand-crafted behaviour by 6.2, 10.6, and 19.9 system turns for the different distributions of confidence levels, respectively (from top to bottom). This result suggests that the importance of machine dialogue optimization grows according to the size of the conversational agent.

It can be noted that the HAM-based policy did better than the fully-learnt policy. This is presumably due to the following reasons: (1) that the fully-learnt policy uses the whole action set, and explores incoherent actions; and (2) that the fully-learnt policy exhibited infinite loops during testing, meaning that during training it takes longer to reach the goal states. Test results showed that the HAM-based policies also exhibited dialogues with infinite loops, e.g. the policy eventually apologized infinitely often.

[2] In a similar way to the flight booking system, learning was frozen in the first one hundred dialogues and the Q-values were initialized to ‘1’ for state-actions pairs that matched the hand-crafted behaviour. This setting was employed to observe the policy’s performance before learning.

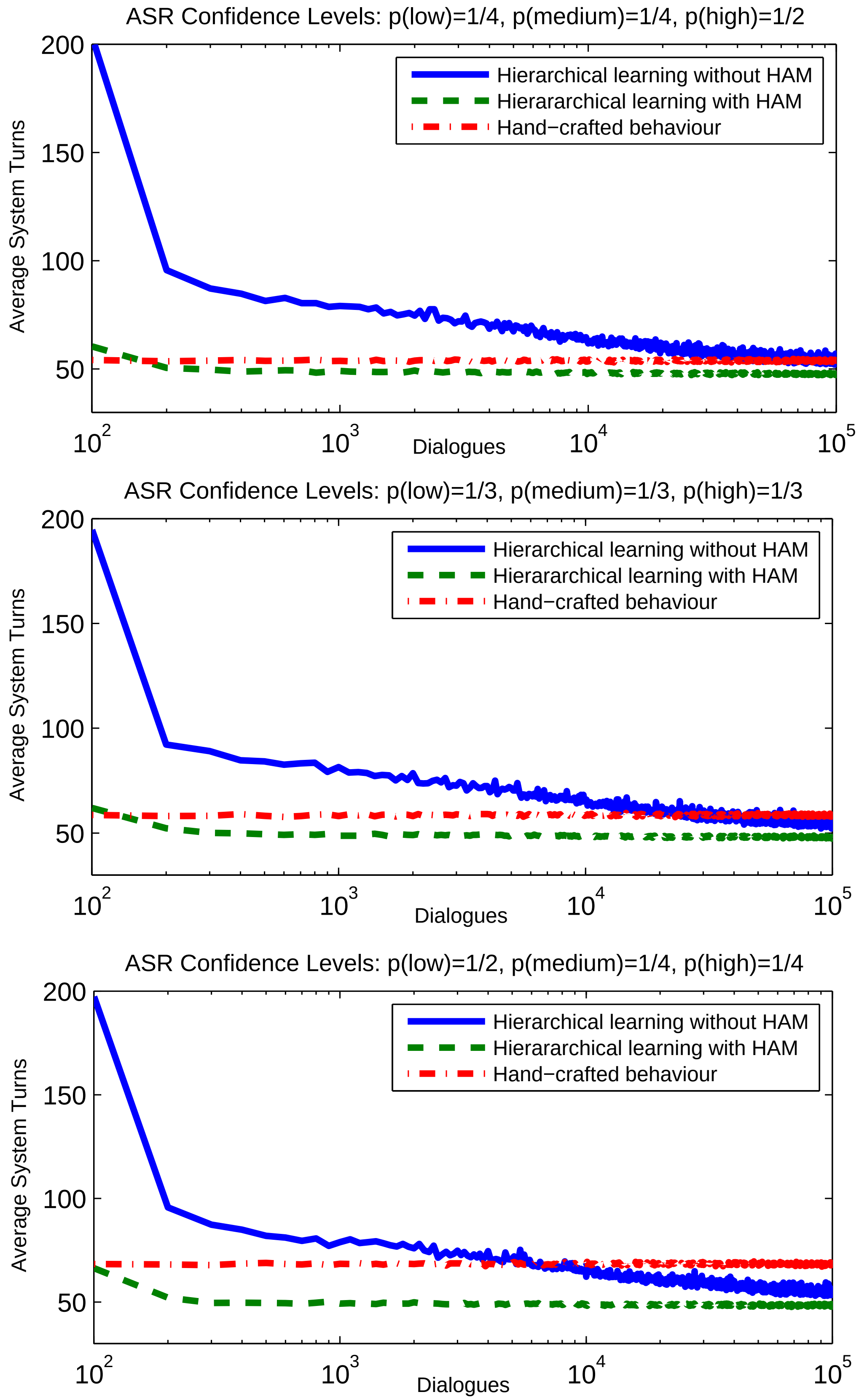


Figure 6.11: *Learning curves of dialogue policies in the 26-slot travel planning system using the reward function defined by equation 5.9. In the last 10^4 dialogues the HAM-based policy averaged* 6.2, 10.6, *and* 19.9 *fewer system turns than hand-crafted behaviour for the different distributions of confidence levels (from top to bottom).*

The issue of infinite dialogues motivated us to constrain further the HAM-based policy of Figures 6.8 and 6.9, and to use the reward function defined by equation 5.10. On the one hand, such a reward function penalized strongly an action that did not change the current dialogue state. On the other hand, the additional constraints in the HAM consisted in prohibiting apologies in medium and high confidence levels, which resulted in a more compact state-action space of 160871 state-actions. This represents a relative reduction of $\sim 80\%$ state-actions compared to the hierarchical state-action space without HAM. Figure 6.12 shows the learning curves for this more compact HAM-based dialogue policy (with frozen learning in the first 100 dialogues) for different amounts of ASR confidence levels, averaged over 10 training runs of 10^5 dialogues.

An evaluation on the last 10^4 dialogues reports that the HAM-based policy with further constraints solved the the problem of dialogues with infinite loops. This result tells us that learning with prior knowledge provides a framework to specify constraints on the solution. In addition, it was observed that the reward function defined by equation 5.10 generated more efficient dialogues in the HAM-based policy (on average 1.5 system turns), also referred to as 'semi-learnt behaviour' (see Table 6.3). The HAM-based policy outperformed the hand-crafted one by 7.6, 12.1, and 21.4 system turns for each distribution of confidence levels, respectively. These gains in dialogue efficiency also highlight the importance of validating these results in a realistic environment. An evaluation with real users of learnt dialogue policies derived from equation 5.10 and balanced ASR confidence levels is reported in chapter 7.

All these results make the combined hierarchical learning approach more attractive for application in real-world spoken dialogue systems, and the learning efficiency of this approach is attractive for optimizing dialogue behaviour in an online setting.

Table 6.3: *Average system turns of policies in the last 10^4 training dialogues, where the third column used the reward function described by equation 5.9 and the fourth column used the reward function described by equation 5.10.*

Confidence Level Distribution (low, medium, high)	Hand-crafted Behaviour	Semi-learnt Behaviour[1]	Semi-learnt Behaviour[2]
Distribution1 $(1/4, 1/4, 1/2)$	53.8 ± 0.9	47.6 ± 0.7	46.2 ± 0.7
Distribution2 $(1/3, 1/3, 1/3)$	58.6 ± 1.0	48.0 ± 0.8	46.5 ± 0.8
Distribution3 $(1/2, 1/4, 1/4)$	68.3 ± 1.4	48.5 ± 0.9	46.9 ± 0.8

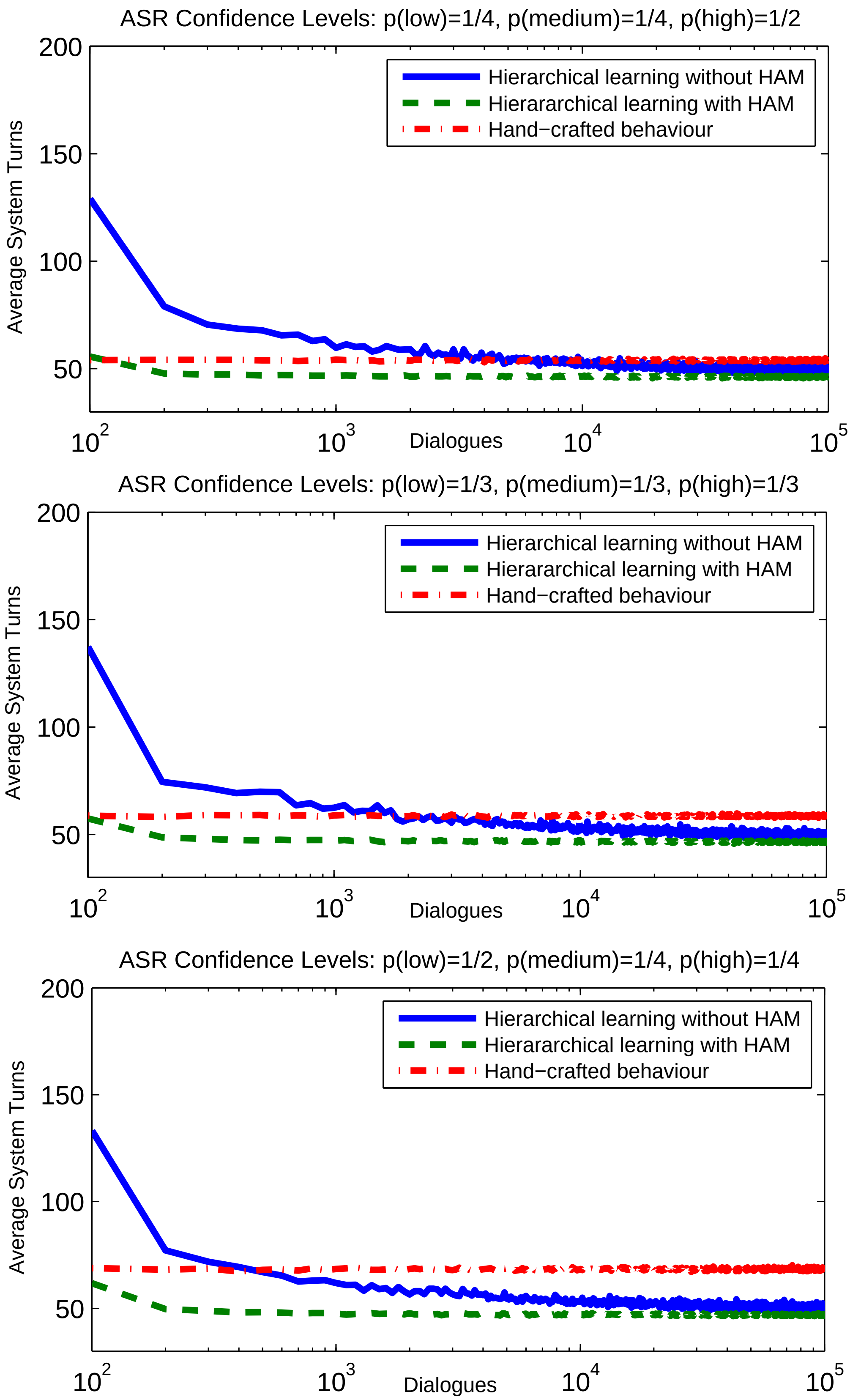


Figure 6.12: *Learning curves of dialogue policies in the 26-slot travel planning system using the reward function defined by equation 5.10. In the last* 10^4 *dialogues the HAM-based policy averaged* 7.6, 12.1, *and* 21.4 *fewer system turns than hand-crafted behaviour for the different distributions of confidence levels (from top to bottom).*

6.4.4 Analysis of learnt behaviours with finite dialogues

This sub-section analyzes the performance of learnt policies without infinite loops. Table 6.4 shows test results for hand-crafted and semi-learnt behaviour (using the more compact HAM) also averaged over 10 test runs of 1000 dialogues. It can be noted that the semi-learnt dialogue behaviour outperformed the hand-crafted one by 10%, 16% and 28% fewer system actions for each confidence level distribution, respectively. This reduction of system actions can be briefly explained as follows: the semi-learnt behaviour differs from the hand-crafted one in the use of more acceptances (action 'acc'), more multiple implicit confirmations (action 'mic'), fewer apologies (actions 'apo+req' and 'apo+ofr'), and fewer multiple explicit confirmations (action 'mec').

Table 6.4: *Test results showing the average number of primitive actions per dialogue of semi-learnt policies with different amounts of ASR confidence levels (low, medium, high). The number of actions per dialogue (in bold) within each ASR confidence level distribution were compared with t-tests and showed statistical significance at* $p < 0.01$.

Conf. Levels	$(1/4, 1/4, 1/2)$		$(1/3, 1/3, 1/3)$		$(1/2, 1/4, 1/4)$	
Action	Hand-crafted	Semi-Learnt	Hand-crafted	Semi-Learnt	Hand-crafted	Semi-Learnt
acc	3.61	3.93	2.82	3.81	2.85	3.30
ack	4.02	4.03	4.02	4.03	4.02	4.02
apo+ofr	1.18	0.37	1.80	0.00	3.47	0.00
apo+req	4.53	0.59	7.23	0.67	14.23	0.94
clo	1.00	1.00	1.00	1.00	1.00	1.00
dbq+sta	4.47	4.47	4.47	4.47	4.46	4.47
gre	1.00	1.00	1.00	1.00	1.00	1.00
mec	5.84	4.32	6.87	4.32	7.63	4.13
mic+req	2.35	2.82	1.92	2.77	1.99	2.83
ofr	0.27	0.27	0.26	0.27	0.26	0.26
pre+ofr	4.02	4.03	4.02	4.03	4.02	4.02
rel	0.47	0.46	0.48	0.47	0.46	0.47
req	10.63	10.11	11.52	10.23	11.66	10.68
sec	10.09	10.24	10.55	10.37	10.46	11.00
sic+req	3.15	3.07	2.73	2.99	2.71	2.36
Sum	**56.63**	**50.69**	**60.68**	**50.42**	**70.24**	**50.50**

Note that the previous differences in dialogue efficiency are smaller than those reported by fully-learnt behaviour without infinite loops (Table 5.6). This motivated us to test again the same semi-learnt behaviour, but acting according to eq. 5.11. It was found that doing this helped to generate more efficient conversations: the semi-learnt behaviour outperformed the hand-crafted one by 16%, 22% and 32% fewer system actions for each confidence level distribution, respectively (Table 6.5). These differences in dialogue efficiency are more comparable to those obtained by fully-learnt behaviour (see section 5.5.3). These results suggest that fully- and semi-learnt behaviours can perform comparably on finite dialogues, and that a combination of hand-crafted and (semi) learnt policies may result in better performance than using them separately.

Table 6.5: *Test results showing the average number of primitive actions per dialogue of semi-learnt policies (acting according to eq. 5.11) with different amounts of ASR confidence levels. The number of actions (in bold) within each confidence level distribution were compared with t-tests and showed statistical significance at* $p < 0.01$.

Conf. Levels	$(1/4, 1/4, 1/2)$		$(1/3, 1/3, 1/3)$		$(1/2, 1/4, 1/4)$	
Action	Hand-crafted	Semi-Learnt	Hand-crafted	Semi-Learnt	Hand-crafted	Semi-Learnt
acc	3.61	3.97	2.82	3.89	2.85	3.79
ack	4.02	4.02	4.02	4.02	4.02	4.03
apo+ofr	1.18	0.30	1.80	0.06	3.47	0.10
apo+req	4.53	0.51	7.23	0.57	14.23	0.90
clo	1.00	1.00	1.00	1.00	1.00	1.00
dbq+sta	4.47	4.47	4.47	4.48	4.46	4.48
gre	1.00	1.00	1.00	1.00	1.00	1.00
mec	5.84	2.16	6.87	2.21	7.63	2.13
mic+req	2.35	2.82	1.92	2.80	1.99	2.88
ofr	0.27	0.25	0.26	0.25	0.26	0.25
pre+ofr	4.02	4.02	4.02	4.02	4.02	4.03
rel	0.47	0.44	0.48	0.45	0.46	0.45
req	10.63	9.73	11.52	9.82	11.66	9.93
sec	10.09	9.62	10.55	9.74	10.46	9.90
sic+req	3.15	3.03	2.73	2.96	2.71	2.77
Sum	**56.63**	**47.36**	**60.68**	**47.27**	**70.24**	**47.63**

6.5 Related work

Our approach for incorporating prior expert knowledge into reinforcement learning agents is based on the Hierarchical Abstract Machines (HAMs) of (Parr and Russell, 1997). In this approach the system designer specifies a partial program (HAM) and leaves the unspecified part to the hierarchical reinforcement learning agent.

Litman et al. (2000); Singh et al. (2002) incorporated prior knowledge into an MDP-based dialogue system (NJFun) by means of hand-crafted rules used to compress the state-action space. This approach allowed them to perform very efficient learning. Our approach differs from Litman and co-workers' approach in two respects: (1) NJFun does not provide a formal framework to incorporate prior knowledge, our approach is based on deterministic-stochastic finite state machines; (2) NJFun applies flat dialogue optimization, while our approach applies hierarchical optimization.

Heeman (2007) proposed combining the information-state update approach with reinforcement learning dialogue systems. In this approach the information-state (dialogue state) is hand-crafted by update rules based on preconditions and effects. In this combined approach a subset of preconditions that are easy to specify are hand-crafted, and those less easy to specify are left to the reinforcement learning agent. Our approach differs from the Heeman's approach as follows: (1) prior knowledge is specified with deterministic-stochastic finite state machines instead of information-state update rules, and (2) we optimize hierarchical dialogue strategies instead of flat dialogue strategies.

Williams (2008a,b) proposed executing a Partially Observable MDP (POMDP) and a Hand-Crafted (HC) dialogue controller in parallel. At each time step, the HC controller is in state s (e.g. semantic frame) and the POMDP is in belief state b (probability distribution over POMDP states), the HC controller nominates a subset of actions, and the POMDP updates a value function only for that particular subset of actions. Thus, a POMDP solution is found on a more compact space of policies. Our approach and Williams's approach share the idea of executing a partial program in parallel with an optimized decision-making model, but they differ as follows: (1) our HAM-based form of prior knowledge does not consider belief states: nevertheless, HAMs can be used for decision-making at higher levels where dialogue states can be identified with certainty; (2) whilst the HC controller of Williams's approach is an arbitrary computer program, our approach is based on deterministic-stochastic finite state machines and provides a formal reinforcement learning method; and (3) our approach optimizes a hierarchy of partial programs, which is more scalable and suitable for reusability.

6.6 Discussion

This chapter addresses further issues in dialogue optimization for real world systems: (a) the role of prior, or expert knowledge, (b) sub-optimal solutions, (c) search space reduction before learning, (d) reusable solutions, and (e) partially specified behaviour.

Firstly, the approach of learning dialogue policies without prior knowledge simply exacerbates the problem. The role of prior knowledge is important for at least two reasons: (1) to reduce the search space in order to find faster solutions and with reduced computational demands, and (2) to allow the opportunity to incorporate ad hoc constraints due to system requirements. In addition, the incorporation of prior knowledge in reinforcement learning for spoken dialogue systems has proved to be very useful in order to optimize behaviour from real conversations (Walker, 2000; Litman et al., 2000; Singh et al., 2002). Otherwise, a large number of dialogues is required for such a purpose, and this is only possible with simulations (which can be unrealistic).

Secondly, one of the dangers of using prior knowledge in reinforcement learning agents is that sub-optimal solutions may be obtained. Therefore, the quality of the learnt policies using the approach proposed in this thesis (divide and conquer plus prior knowledge) will depend on two aspects: the hierarchical state representation, and the HAM-based partially specified dialogue strategy. Nevertheless, the learning agents will find optimal context-independent policies according to the specified constraints.

Thirdly, a typical approach to incorporate prior knowledge into reinforcement learning agents is to reduce state-action pairs before learning. This represents a problem if spoken dialogue behaviours are frequently updated (either restricting it or extending it), where a new learnt policy has to be found. This is a strong reason for preferring hand-crafted instead of learnt dialogue behaviour. The proposed partial programs can avoid re-learning when additional deterministic behaviour is incorporated into the HAMs. But the policies must be re-learnt when additional stochastic behaviour is incorporated into the HAMs. In general, partial programs running in parallel with learnt behaviour may help to reduce re-learning of policies with frequent updates.

Fourthly, a desirable property in approaches for building spoken dialogue systems is that of reusable components. The topic of reusable learnt dialogue behaviours is important for at least two reasons. First, it aims to relieve system developers of the effort of doing many expert tasks. Second, it aims to speed up the development-deployment process for conversational agents. In this context, the application of hierarchical SMDPs to dialogue systems may become relevant for the following reasons: (1)

by reusing learnt dialogue behaviours like those generated from chapter 5, and (2) by reusing modularized prior expert knowledge as proposed in this chapter. Lemon et al. (2006a) reuse a single policy (exactly the same) in different dialogue contexts. However, there is much more to do – such as reusing similar behaviours – for facilitating the rapid development of conversational agents with optimized behaviours.

Fifthly, the idea of partially specified dialogue strategies is relevant to the field because it is useful to balance the strengths of purely learnt behaviour and purely hand-crafted behaviour. The approach of Levin and Pieraccini (1997); Levin et al. (2000) is to design automatically the behaviour of dialogue systems. This thesis argues that semi-learnt behaviour is more attractive for the following reasons: (a) it is more coherent than purely learnt behaviour, (b) it plays a more active role in the system's development life cycle, and (c) it is more suitable for online learning.

Finally, two approaches have been proposed in this thesis: (1) SMDP-based hierarchical dialogue optimization, and (2) SMDP-based hierarchical dialogue optimization constrained with HAMs. These approaches complement each other in order to provide a more scalable and flexible composite approach for optimizing spoken dialogue agents. The next chapter describes an experimental evaluation with real users.

6.7 Conclusions

This chapter proposed learning partially specified dialogue strategies using constrained Semi-Markov decision processes and hierarchical reinforcement learning. These partial strategies are specified through hierarchical abstract machines, where obvious behaviour is specified with deterministic choices and non-obvious behaviour with stochastic choices. The latter is the behaviour to be learnt by the reinforcement learning agent. It was applied experimentally to simulated dialogue systems in the flight booking and travel planning domains , and the proposed approach was compared with reinforcement learning ab initio. Experimental results show that the flight booking system used only 0.20% of the flat state-action space, and the travel planning system less than 0.01% of the flat state-action space. Hence learning is much faster and with less computational demands than learning without prior knowledge. Even with such reductions, the learnt dialogue policies outperformed hand-crafted behaviour. In addition, it was found that a combination of hand-crafted and (semi) learnt policies may result in better performance than using them separately. All these results suggest that the proposed approach can be applied to large-scale and real-world spoken dialogue systems.

Chapter 7

A spoken dialogue system using hierarchical reinforcement learning

This chapter aims to validate the hypotheses and preliminary conclusions derived from the previous chapters. Section 7.1 explains the need for more sophisticated spoken dialogue systems. Section 7.2 describes the architecture of a travel planning spoken dialogue system with three different dialogue behaviours: deterministic, fully-learnt, and semi-learnt. The first is used as a baseline for the latter two that employ spoken dialogue strategies generated by hierarchical reinforcement learning. Section 7.3 reports on a quantitative and qualitative evaluation in a laboratory setting with real users. Section 7.4 discusses the strengths and weaknesses of the spoken dialogue system under evaluation. Finally, section 7.5 provides a summary of findings.

7.1 Introduction

The behaviour of spoken dialogue systems is typically hand-coded by designers and developers. This approach has several limitations: it is prone to errors, time-consuming, ad hoc, non-optimized, and non-adaptive, among others. A potential solution is systems that learn their dialogue behaviour (Levin and Pieraccini, 1997) through the use of some sort of intelligent agent that behaves rationally during the dialogue by choosing the best actions according to some performance measure (Russell and Norvig, 2003). Zue (2007) proposed a long-term vision of dialogue systems that can learn, grow, and reconfigure themselves. See chapter 2 for a brief literature review on spoken dialogue systems that learn their dialogue behaviour using reinforcement learning.

Briefly, previous research in dialogue strategy design using the reinforcement learning paradigm has been carried out through two types of conversational environment: real and simulated. Performing experiments on real environments requires large amounts of time, effort and resources. This explains the relative lack of investigations in the field, where two approaches have been employed: first, learn behaviour from real dialogues and then test it on a real environment (Walker, 2000; Singh et al., 2002); and second, learn behaviour on a simulated environment and then test it on a real one (Lemon et al., 2006a; Young et al., 2007; Toney, 2007). The former may be referred to as 'real learnt behaviour', the latter as 'simulated learnt behaviour'. On the one hand, real learnt behaviour is more attractive because it uses real data, but it is not very practical due to the large number of dialogues required for optimal learning. On the other hand, simulated learnt behaviour is more practical but the right things may not be learnt due to the use of a simulated environment, which will inevitably be simpler than the real environment. This suggests that both behaviours have to be backed up with testing on real environments to guarantee their performance.

The problem addressed here is the evaluation of learnt behaviours for large-scale spoken dialogue systems. Most previous investigations of learned spoken dialogue behaviours have been concerned with evaluating small-scale systems, typically using a single dialogue goal with few slots of information. This limitation was the motivation to propose and evaluate a more scalable dialogue optimization framework. The idea of evaluating learnt dialogue behaviours with real users is particularly relevant for showing the effectiveness of the proposed dialogue simulation environment, and the hierarchical reinforcement learning framework described in chapters 5 and 6. For this purpose a heuristic-based simulation framework was used to generate human-machine conversations, producing coherent and distorted conversations (see chapter 4). Once the learning agents designed the dialogue behaviours, they were put into operation in a realistic environment, in the domain of travel planning. The resulting spoken dialogue system allowed users to book flights, hotels and cars. This system shares similarities with the DARPA Communicator dialogue systems (Walker et al., 2002), but used dialogue behaviours designed by hierarchical reinforcement learning agents, using the Semi-Markov decision processes formalism.

The objectives in this chapter were to show that the proposed dialogue simulator can help learning agents to find dialogue strategies that outperform hand-coded, deterministic behaviour, and that hierarchical dialogue behaviours learnt in the presence of constraints derived from prior knowledge (**semi-learnt behaviours**) are more suited to deployment than fully deterministic or fully-learnt dialogue behaviours.

7.2 System architecture

The CSTR travel planning spoken dialogue system supported deterministic or learnt dialogue behaviour. The latter uses dialogue strategies designed by hierarchical reinforcement learning agents on a simulated environment (see chapters 4-6). This system is based on the Open Agent Architecture (OAA) (Cheyer and Martin, 2001). Figure 7.1 shows a high-level architecture using eight OAA-based agents in order to support speech-based task-oriented human-machine communication. The communication flows between facilitator (parent) and the other agents (children). Briefly, the user gives speech signals x_t^u corresponding to words w_t^u, concepts or slots c_t^u, and dialogue acts a_t^u. However, the machine understands them with distortions ($\tilde{w}_t^u$, $\tilde{c}_t^u$, $\tilde{a}_t^u$), and answers back to the user with speech signals x_t^m corresponding to words w_t^m, slots c_t^m, and dialogue acts a_t^m. The user may also misunderstand the machine, and so on until one of the conversants terminates the conversation at T system turns. The rest of this section describes each agent based on dialogue fragments showing inputs and outcomes.

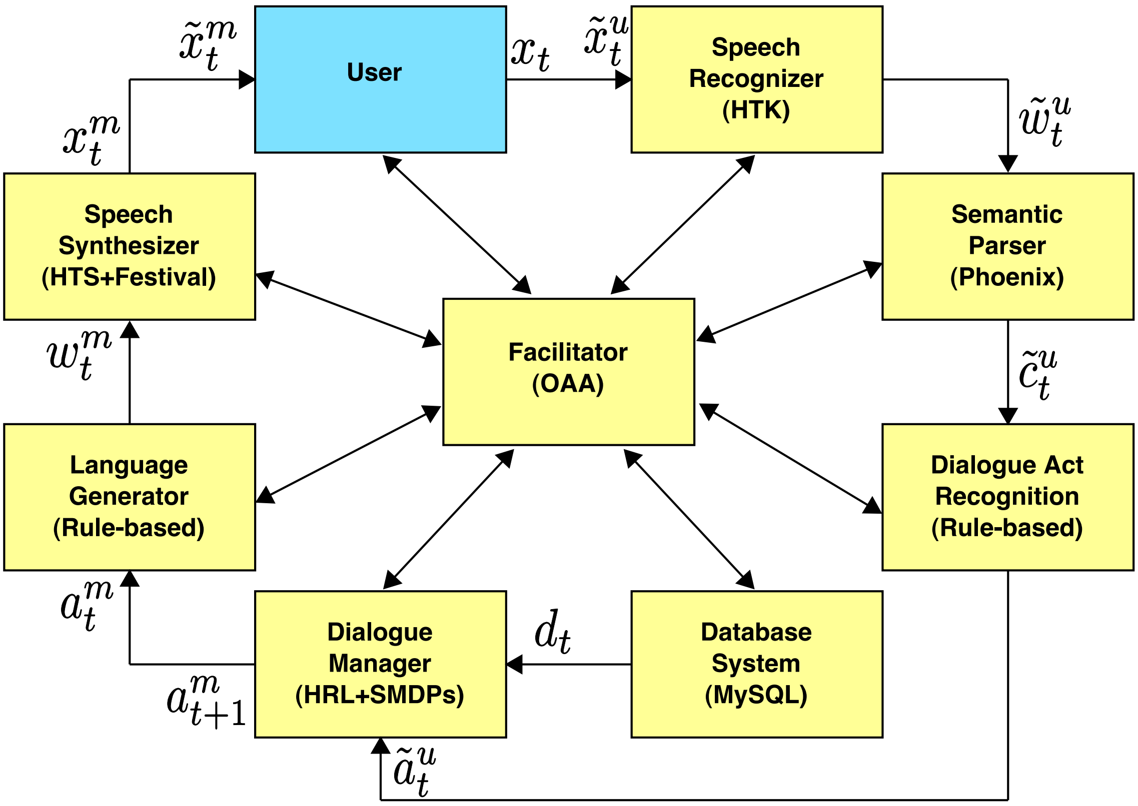


Figure 7.1: *Architecture of the CSTR travel planning spoken dialogue system supporting deterministic or learnt dialogue behaviour. Human-machine communication is carried out with speech signals x_t, words w_t, concepts or slots c_t, and dialogue acts a_t.*

7.2.1 Facilitator agent

OAA is an agent-based framework to build autonomous, flexible, fault-tolerant, distributed and reusable software systems (Cheyer and Martin, 2001). OAA agents can be written in multiple programming languages and run on a computer network with different operating systems. They have a parent agent called *facilitator*, coordinating the communication of child agents by keeping a knowledge base of their services. Child agents are service providers and service requesters. The former let the facilitator know of their own capabilities, and the latter request capabilities from other agents. They communicate by passing string messages between child agents and facilitator.

7.2.2 Speech recognition agent

The task of this agent was to receive user speech signals after each machine prompt w_t^m and to generate a word sequence including confidence levels $\tilde{w}_t^u$, derived from the recognition hypothesis incorporating confidence scores $\bar{w}_t^u$. This agent used the multithreaded ATK API, which is a layer on top of the HTK speech recognition libraries (Young, 2007, 2006). This agent used the acoustic models (trained with data from British speakers) generated from the TALK project[1], and customized-based language models with a lexicon of 263 words. The confidence levels were assigned by dividing the confidence score range $[0...1]$ into three equal areas, equivalent to l =low, m =medium, and h =high confidence. The following table illustrates this process.

ID	Event	Outcome
w_t^m	Machine prompt	Welcome to the CSTR travel planning system. Tell me your flight information.
w_t^u	User response	I would like a single flight from Edinburgh to Paris.
$\bar{w}_t^u$	ASR hypothesis with confidence scores	how(0.27) about(0.31) a(0.15) single(0.60) flight(0.56) with(0.32) b._m._i.(0.47) from(0.70) edinburgh(0.59) to(0.40) paris(0.56)
$\tilde{w}_t^u$	ASR hypothesis w/conf. levels	how(l) about(l) a(l) single(m) flight(m) with(l) b._m._i.(m) from(h) edinburgh(m) to(m) paris(m)
w_{t+1}^m	Machine prompt	A single flight from Edinburgh to Paris. travelling with BMI. When do you want to travel? ...
w_{t+1}^u	User response	I would like to travel with Air France.

[1]Our ASR and TTS agents used wrappers generated from the TALK project (Lemon et al., 2005).

7.2.3 Semantic parsing agent

This agent generated concept or keyword sequences $\tilde{c}_t^u$ from a (distortedly) recognised word sequence $\overline{w}_t^u$. This agent used the Phoenix spontaneous speech parser that maps a word string into a semantic frame. A semantic frame is a set of slots of information, each slot with an associated context-free grammar. Such grammars are compiled into recursive transition networks, which are matched with the given word sequence by a top-down chart parsing algorithm (Ward, 1994). This agent used 3 frames (corresponding to flights, hotels and cars) including 18 semantic networks. See the table below for a sample parsed word sequence.

ID	Event	Outcome
w_t^m	Machine prompt	Welcome to the CSTR travel planning system. Tell me your flight information.
w_t^u	User response	I would like a single flight from Edinburgh to Paris.
$\overline{w}_t^u$	ASR hypothesis wo/conf. scores	how about a single flight with b._m._i. from edinburgh to paris
$\tilde{c}_t^u$	Semantic parse	Flight:[FlightType].SINGLE Flight:[DepCity].[City].EDINBURGH Flight:[DesCity].[City].PARIS Flight:[Airline].BMI
$\tilde{a}_t^u$	User dialogue act	pro(FlightType=single.m,DepCity=edinburgh.m, DesCity=paris.m,Airline=bmi.m)
w_{t+1}^m	Machine prompt	A single flight from Edinburgh to Paris travelling with BMI. When do you want to travel? ...
w_{t+1}^u	User response	I would like to travel with Air France.

7.2.4 Dialogue act recognition agent

This agent generated user dialogue acts $\tilde{a}_t^u$ using a two-stage approach. First, a user dialogue act type was selected taking into account the current concept sequence $\tilde{c}_t^u$ and last machine dialogue act corresponding to the machine prompt w_t^m. Once a dialogue act type had been selected, it took context into account to become a user dialogue act $\tilde{a}_t^u$. Although it is possible to generate more than one dialogue act per user utterance, this agent generated a single user dialogue act (see the table above for an example).

7.2.5 Database system agent

This agent returned database tuples based on SQL queries from the dialogue manager. It used a web scraper to populate a local database, retrieving travel data from a commercial web site (`www.opodo.co.uk`). This strategy was selected in order to avoid long time responses from direct queries to the web site. The database stored information from dialogue scenarios such as the shown in Table 7.2. This agent returned an empty data set when it experienced unknown slot-values. That situation occurred if some slots were filled incorrectly due to mis-recognitions, which forced the user to reprovide the wrong information in order to obtain a non-empty data set (see example below). That situation also served to test the early subtask termination (section 5.2.4).

```
SYS: Welcome to the CSTR travel planning system.
     Tell me your flight information.
USR: I would like a single flight from Edinburgh to Paris.
SYS: A single flight from Edinburgh to Paris travelling with BMI.
     When do you want to travel?
USR: I would like to travel with Air France
SYS: A flight travelling with Air France. When do you want to travel?
USR: [cut off] the sixth
SYS: A flight leaving on May sixth.
     At what time do you prefer to travel?
USR: Ehh I would like to travel on December the sixth in the morning
SYS: At what time do you prefer to travel? For example you can say:
     A flight in the afternoon.
USR: A flight in the morning
SYS: Did you say a flight in the morning?
USR: Yes
SYS: Please wait while I retrieve information. I am sorry I didn't
     find any flights from Edinburgh to Paris on may sixth travelling
     with Air France. Try again with some different information.
USR: December the sixth
SYS: Did you say a single flight leaving on December sixth
USR: Yes
SYS: Please wait while I retrieve information. I found 2 flights from
     Edinburgh to Paris with Air France...
```

7.2.6 Dialogue management agent

This agent generated machine dialogue acts a_t^m from the hierarchy of policies π_j^i based on three different types of dialogue behaviours: deterministic, fully-learnt and semi-learnt. The *deterministic* strategy is based on a mapping from states to actions $f(s,a)$ using the baseline machine dialogue behaviour described in section 4.3.2:

$$\pi_j^i(s) = f(s,a), \tag{7.1}$$

where s is a state in S_j^i and a is an action in A_j^i. The two learnt behaviours were optimized for efficient dialogues. The *fully-learnt* behaviour used a hierarchical dialogue agent (based on eq. 5.10) that could use the full action set in each state (see chapter 5):

$$\pi_j^{*i}(s) = \arg\max_{a \in A_j^i} Q_j^{*i}(s,a). \tag{7.2}$$

In contrast, the *semi-learnt* behaviour (also based on eq. 5.10) used "partially specified dialogue strategies" for constraining the actions in each joint state $w = (s, \overline{s})$, where s is an environment state and $\overline{s}$ is a choice state in the partial policy (see chapter 6):

$$\pi_j^{*i}(w) = \arg\max_{a \in A_j'^i} Q_j^{*i}(w,a). \tag{7.3}$$

Table 7.1 shows an example of the form of dialogue control in the CSTR travel planning system given by a hierarchical reinforcement learning agent with fully-learnt behaviour. The agent uses a hierarchy of learnt dialogue policies π_j^{*i}, where each policy chooses the action with the highest cumulative reward for each state. Notice that machine decisions can be primitive actions such as {'req=request', 'mic=multiple implicit confirmation'}, or composite actions (also referred to as 'subtasks') such as {'M_0^2=sub-dialogue for outbound flight', 'M_0^3=sub-dialogue for collecting mandatory slots in the outbound flight'}. A dialogue subtask uses a separate learnt policy to act in the sub-dialogue. When a subtask is invoked, it obtains its initial dialogue state from the machine's knowledge base that is updated from observations in the environment. A subtask returns to its parent subtask when it reaches a terminal state.

The specifications of these spoken dialogue controllers were expressed in XML files encapsulating information about hierarchical dialogue structures, learning environment, knowledge bases of simulated user and machine, and learning setup (see chapters 4, 5, 6, and appendix B for more information). Since these spoken dialogue behaviours only differ in their action-selection mechanism, and the rest of the OAA-based agents (see figure 7.1) did not change regardless of the behaviour of choice, it is fair to say that these behaviours were evaluated under similar conditions.

Table 7.1: *Fragment of a real dialogue in the CSTR travel planning system using policies π_j^{*i}, the state representation is shown in Table 5.2.*

Agent	Policy	State	Action	Dialogue Act	Utterance
SYS	π_0^{*0}	0,0,0,0,0,0	gre	gre()	Welcome to the CSTR travel planning system.
		0,1,0,0,0,0	M_0^1		
	π_0^{*1}	1,0,0	M_0^2		
	π_0^{*2}	0,0,0,0,0	M_0^3		
	π_0^{*3}	0,0,0,0,0,0,0	req	req(DepCity)	Tell me your flight information.
USR				pro(FlightType=single,DepCity=edinburgh, DesCity=paris)	*I would like a single flight from Edinburgh to Paris.*
SYS	π_0^{*3}	0,2,2,0,0,2,2	acc		
	π_0^{*3}	2,2,2,0,0,2,2	mic	mic(FlightType=single,DepCity=edinburgh, DesCity=paris,Airline=bmi)+req(DepDate)	A single flight from Edinburgh to Paris travelling with BMI. When do you want to travel?
USR				pro(Airline=air france)	*I would like to travel with Air France.*
SYS	π_0^{*3}	2,4,4,0,0,2,4	sic	sic(Airline=air france)+req(DepDate)	A flight travelling with air france. When do you want to travel?
USR				pro(DepDate=december sixth)	*December the sixth*
SYS	π_0^{*3}	2,4,4,2,0,4,4	acc		
SYS	π_0^{*3}	3,4,4,2,0,4,4	sic	sic(DepDate=may sixth)+req(DepTime)	A flight leaving on May sixth. At what time do you prefer to travel?
USR				rep(DepDate=december sixth, DepTime=morning)	*ehh i would like to travel on December the sixth in the morning.*
...					
SYS	π_0^{*3}	3,4,4,4,4,4,4			[terminal state]
	π_0^{*2}	0,0,2,2,0	dbq+sta	dbq(db)+sta(db)	Please wait while I retrieve information...

7.2.7 Language generation agent

The task of this agent was to generate a machine prompt w_t^m in natural language based on a template-based approach. A prompt template has a word sequence embedding variables, and was selected given the current machine dialogue act a_t^m, dialogue state s_t^m or joint state w_t^m, and a simple help mechanism[2]. Once a prompt template had been selected, it took context into account by replacing variables with values in the machine's knowledge base in order to generate the word sequence w_{t+1}^m. This agent included 463 prompt templates. The table below (with omitted dialogue states) shows a sample prompt template c_t^m and its corresponding machine prompt w_{t+1}^m.

ID	Event	Outcome
w_t^m	Machine prompt	Welcome to the CSTR travel planning system. Tell me your flight information.
$\tilde{a}_t^u$ a_t^m	User dialogue act Machine Dialogue act	pro(FlightType=single.m,DepCity=edinburgh.m, DesCity=paris.m,Airline=bmi.m) mic(FlightType=single,DepCity=edinburgh, DesCity=paris,Airline=bmi)+req(DepDate)
c_t^m	Prompt for action 'mic' Prompt for action 'req'	A $FlightType flight from $DepCity to $DesCity travelling with $Airline. When do you want to travel?
w_{t+1}^m	Machine prompt	A single flight from Edinburgh to Paris travelling with BMI. When do you want to travel? ...
w_{t+1}^u	User response	I would like to travel with Air France.

7.2.8 Speech synthesis agent

This agent generated speech signals x_t^m from a given word sequence w_t^m. This agent is based on the Festival text-to-speech system[3] with an HTS voice generated from eight hours of recorded speech (Yamagishi et al., 2007). The speech signals were generated online, using a pre-processing stage to split word sequences at punctuation symbols in order to avoid long silences in the machine's utterance.

[2]Simple automatic help: a) 1st slot collection=no help, b) 2nd collection=help prompt suggesting to fill multiple slots, c) 3rd collection: help prompt suggesting a shorter sentence, d) 4rd collection=help prompt suggesting to fill a single slot, e) others=help prompt suggesting to rephrase the sentence.

[3]`http://www.cstr.ed.ac.uk/projects/festival`

7.3 System evaluation

These experiments aimed to investigate whether hierarchically learnt dialogue behaviour can outperform deterministic behaviour in a realistic environment, and to evaluate the heuristic simulation environment with real data. For such a purpose the system described in the previous section was implemented and deployed to a population of real users for its corresponding evaluation. See appendix C for a sample dialogue.

7.3.1 Evaluation methodology

The CSTR travel planning spoken dialogue system was evaluated using a number of metrics, mostly derived from the PARADISE framework (Walker et al., 2000), which has been widely accepted for evaluating the performance of spoken dialogue systems.

(i) **Dialogue Efficiency**: This group of quantitative metrics includes *system turns*, *user turns*, and *elapsed time* (in seconds). All of them report averages per dialogue goal (flight, hotel, car). Elapsed time includes the time used by both conversants.

(ii) **Dialogue Quality**: This group of metrics includes *Word Error Rate* (WER), *Keyword Error Rate* (KER), and *Event Error Rate* (EvER). The latter is decomposed into the following metrics reported as percentages: *correct acceptance*, *correct confirmation*, *correct rejection*, *false acceptance*, *false confirmation* and *false rejection*. Other commonly reported metrics include percentages of commands and barge-ins, but this dialogue system did not support them.

(iii) **Task Success**: This group of quantitative metrics includes *task success* and *dialogue reward*. Task success uses a binary approach, where each dialogue task is classified as successful if the user achieved the goal (e.g. booking a flight, hotel or car) as in (Bohus and Rudnicky, 2005b). Dialogue reward combines task success and dialogue length in terms of system turns (Lemon et al., 2006a):

$$DialogueReward = \begin{cases} 100 \text{ - } |\text{SystemTurns}| & \text{for} \quad \text{successful dialogue} \\ 0 \text{ - } |\text{SystemTurns}| & \text{for} \quad \text{failed dialogue} \end{cases} \tag{7.4}$$

(iv) **User Satisfaction**: These qualitative metrics include *easy to understand*, *system understood*, *task easy*, *interaction pace*, *what to say*, *system response*, *expected behaviour*, and *future use*. Their sum represents the overall user satisfaction score.

7.3.2 Experimental setup

The experiments of this research were restricted to a user population of native speakers of English and evaluated the three machine dialogue behaviours described in the previous three chapters: deterministic (‘D’), fully-learnt (‘F’), and semi-learnt (‘S’). In these experiments each user was presented with six dialogue tasks (travel bookings), with the system using each of the three behaviours twice, so that each user experienced all behaviours. The first three dialogues concerned single bookings and the last three dialogues concerned composite bookings. Table 7.2 shows examples of single and composite travel booking tasks. The six dialogues per user were collected using one of the following two sequences: DSFFSD and SDFFDS; i.e. half of the users interacted first with a deterministic behaviour, and the other half interacted first with a learnt behaviour. Whilst deterministic and semi-learnt behaviours started the dialogues interchangeably, fully-learnt behaviour always started the composite travel bookings. This sequence of dialogues was used because other alternative sequences such as {DSFFSD, DFSSFD, SDFFDS, SFDFDS, FSDDSF, FDSSDF} require larger data collections (the more data the more expensive and time-consuming).

Table 7.2: *Sample tasks in the CSTR travel planning spoken dialogue system. In the experiments reported here, each user participated in 3 single and 3 composite tasks.*

Booking	Task
Single	Try to book a **single** flight from **London** to **Paris** leaving on **December 6th** in the **afternoon**, and travelling with **any airline**. What is the cost of the most expensive flight?
Composite	a) Try to book a **return** flight from **Edinburgh** to **Amsterdam** leaving on **January 22nd** in the **morning**, and returning on the **1st of February** in the **evening**. What is the cost of the cheapest flight with **British Airways**? b) Try to book a **cheap** hotel in **downtown** with **any hotel** brand. What is the cost of the cheapest hotel in downtown? c) Try to rent a **compact car** near the **airport** for **three days** on **January 22nd** with pick-up time at **7PM**. You don’t have any preference regarding rental company. What is the rental cost of the most expensive car?

Each dialogue was logged using an extended version of the DATE dialogue annotation scheme (Walker and Passonneau, 2001). These log files were used to compute quantitative results. In addition, at the end of each dialogue, participants were asked to fill in a questionnaire (Table 7.3) in order to compute qualitative results, evaluated with a 5-point Likert scale, where 5 represents the highest score.

A population of 32 users voluntarily agreed to participate in the experimental evaluation. They had an average age of 36 with a gender distribution of 69% (22) male versus 31% (10) female. The participants' country of origin were as follows: 53% (17) from the UK, 38% (12) from USA, and 9% (3) from Canada. From this user population, 28% (9) had no experience with spoken dialogue systems, 56% (18) had some experience interacting with a spoken dialogue system at least once, and 16% (5) were expert users. The latter were researchers in spoken dialogue processing.

Table 7.3: *Subjective dialogue measures for qualitative evaluation.*

Measure	Question
Easy to Understand	Was the system easy to understand?
System Understood	Did the system understand what you said?
Task Easy	Was it easy to find the flight/hotel/car you wanted?
Interaction Pace	Was the pace of interaction with the system appropriate?
What to Say	Did you know what you could say at each point?
System Response	Was the system fast and quick to reply to you?
Expected Behaviour	Did the system work the way you expected it to?
Future Use	Do you think you would use the system in the future?

7.3.3 Experimental results

This subsection describes an analysis of results computed from automatic and manual transcriptions at the syntactic and semantic level. Table 7.4 shows a summary of results comparing semi-learnt dialogue behaviour against deterministic and fully-learnt dialogue behaviour; including statistical significance. For such a purpose data vectors (averaged per speaker) were verified through Lilliefors tests which indicated that they do not come from normal distributions. This suggests that non-parametric tests should be used. Thus, significance tests are reported with the Wilcoxon signed-rank test as suggested by (Demsar, 2006).

Table 7.4: *Results of the CSTR travel planning spoken dialogue system comparing three different dialogue behaviours, organized according to the following groups of metrics: dialogue efficiency, dialogue quality, task success and user satisfaction.*

Measure	Deterministic Behaviour$^{(1)}$	Fully-Learnt Behaviour$^{(2)}$	Semi-Learnt Behaviour$^{(3)}$	p-values		
				(1,2)	(1,3)	(2,3)
Avg. System Turns	16.63	12.24	15.09	≤ 0.05	≤ 0.05	≤ 0.05
Avg. User Turns	14.38	9.69	12.63	≤ 0.05	≤ 0.05	≤ 0.05
Avg. Time (secs)	177.23	139.59	165.11	≤ 0.05		
Word Error Rate	0.429	0.410	0.428			
Keyword Error Rate	0.300	0.278	0.301			
Event Error Rate	0.409	0.351	0.372			
Correct Acceptance	5.51	26.34	20.95	≤ 0.05	≤ 0.05	
Correct Confirmation	48.51	36.17	39.86	≤ 0.05	≤ 0.05	≤ 0.05
Correct Rejection	5.18	2.37	1.92	≤ 0.05	≤ 0.05	
False Acceptance	3.25	12.27	9.30	≤ 0.05	≤ 0.05	≤ 0.05
False Confirmation	32.64	20.11	26.60	≤ 0.05	≤ 0.05	≤ 0.1
False Rejection	4.91	2.55	1.36	≤ 0.05	≤ 0.05	
Avg. Task Success	0.94	0.62	0.95	≤ 0.05		≤ 0.05
Avg. Dialogue Reward	79.46	54.68	82.56	≤ 0.05	≤ 0.05	≤ 0.05
Easy to Understand	4.34	4.31	4.44			
System Understood	3.09	2.72	3.28	≤ 0.05		≤ 0.05
Task Easy	3.50	3.00	3.45	≤ 0.1		≤ 0.05
Interaction Pace	3.52	3.55	3.50			
What to Say	3.45	3.47	3.58			
System Response	3.67	3.64	3.63			
Expected Behaviour	3.42	3.08	3.52	≤ 0.05		≤ 0.05
Future Use	3.14	2.83	3.28	≤ 0.05		≤ 0.05
User Satisfaction	28.14	26.59	28.67	≤ 0.1		≤ 0.05

(1) Note on statisfical significance: typically, p-values $p \leq 0.05$ are considered to be statistically significant, and p-values $p \leq 0.1$ are indicative of a statistical trend.

(2) Note on task success: the drop of performance in fully-learnt behaviour was mainly caused by infinite loops, where the execution of action a in state s did not change the state $s' = s$.

7.3.3.1 Analysis of quantitative and qualitative results

Dialogue efficiency: fully-learnt behaviour seems to outperform significantly the other behaviours by obtaining fewer system turns, fewer user turns and less time. This is not surprising because it was known in advance that this dialogue policy included infinite loops in some dialogue states. In the experiments these kind of dialogues were manually stopped after three repetitive actions, considered as evidence of an infinite loop, Table 7.5 shows an example. The purpose of testing this dialogue policy was three-fold: (1) to evaluate how users perceive a dialogue policy with infinite loops; (2) to raise the issue of (in)coherent behaviour inferred by reinforcement learning agents, which has been ignored in previous related work; and (3) to compare its performance against a similar dialogue policy, but constrained with prior expert knowledge.

This phenomenon did not happen with deterministic or semi-learnt behaviours because their prior knowledge constrained more tightly the available actions per dialogue state. From these two dialogue strategies, it can be observed that semi-learnt behaviour outperformed deterministic, with significant differences in system and user turns. These results suggest that although learnt behaviours were optimized for dialogue efficiency, they cannot be evaluated in the same way. Therefore, a wider repertoire of evaluation metrics is preferable for a deeper analysis of dialogue behaviours.

Dialogue quality: Fully-learnt behaviour obtained the lowest word and keyword error rates. These results are not statistically significant, which suggests that behaviours were compared under similar recognition error rates. Deterministic and semi-learnt behaviours exhibited similar word and keyword error rates, but different event error rates. Typically, spoken dialogue strategies have to handle trade-offs between acceptance, confirmation and rejection of recognition hypotheses, which can be classified as correct or incorrect. Ideally, dialogue behaviours should choose actions maximizing correct acceptance/confirmation/rejection events, whilst minimizing the incorrect ones. From these results it can be inferred that semi-learnt behaviour handled such trade-offs better than deterministic behaviour by issuing more acceptances and fewer confirmations. This implies the use of more implicit confirmations and fewer explicit confirmations, which helps to explain why semi-learnt behaviour was more efficient than the deterministic one. Although dialogue policies were not optimized for 'event error rate' (see equation 4.13, p 75), these results suggest that optimizing for dialogue efficiency produced an indirect optimization for such trade-offs. In addition, it can be observed that deterministic and semi-learnt behaviours are significantly different in all

Table 7.5: *Real dialogue with infinite loop in the CSTR travel planning dialogue system, where the fully-learnt policy did not learn the action 'rel=constraint relaxation' for the (mis-)recognized slot values, and kept trying the action 'dbq=database query'.*

Agent	Subtask	State	Action	Utterance
SYS	M_0^0	0,0,0,0,0	gre	Welcome to the CSTR travel planning system.
	M_0^0	0,1,0,0,0,0	M_0^1	
	M_0^1	1,0,0	M_0^2	
	M_0^2	0,0,0,0,0	M_0^3	
	M_0^3	0,0,0,0,0,0,0	req	Tell me your flight information.
USR				I would like to fly from Edinburgh to London.
SYS	M_0^3	0,2,2,0,0,0,0	acc	
	M_0^3	2,2,2,0,0,0,0	mic+	A flight from Edinburgh to London.
			req	When do you want to travel?
USR				On the 6th of December in the evening.
SYS	M_0^3	2,4,4,3,2,0,0	acc	
	M_0^3	4,4,4,3,2,0,0	mic+	A flight on December 6th in the evening.
			req	What is your preferred airline?
USR				-tish Airways
SYS	M_0^3	4,4,4,2,4,0,0	sic+	A flight leaving on May 8th.
	M_0^3		req	What is your preferred airline?
USR				British Airways
SYS	M_0^3	4,4,4,4,4,2,0	acc	
	M_0^3	5,4,4,4,4,2,0	sic+	A flight travelling with British Airways.
			req	What kind of flight do you want?
				Single or return.
USR				Single
SYS	M_0^3	5,4,4,4,4,4,3	sec	Did you say a single flight?
USR				Yes
SYS	M_0^3	5,4,4,4,4,4,4		[terminal state]
	M_0^2	0,0,2,2,0	dbq+sta	Please wait while I retrieve information.
SYS	M_0^2	1,0,2,2,0	dbq+sta	Please wait while I retrieve information.
SYS	M_0^2	1,0,2,2,0	dbq+sta	Please wait while I retrieve information.
...	...	...	...	[manually stopped]

Note: (1) dialogues with 3 repetitive actions were considered as evidence of an infinite loop, and were manually stopped, (2) the state representation is described in Table 5.2.

recognition events (correct/false acceptance/confirmation/rejection). In contrast, both learnt behaviours are significantly different in only half of the recognition events, suggesting that learnt behaviours act in a more similar way than deterministic behaviour.

Task success: Fully-learnt behaviour was significantly outperformed by the other behaviours that generated more successful conversations. This is where fully-learnt behaviour paid the price for generating some infinite dialogues that had to be artificially terminated before successful completion. In addition, whilst deterministic and semi-learnt behaviours were very similar in terms of task success, semi-learnt behaviour significantly outperformed its deterministic counterpart in terms of dialogue reward. This suggests that the dialogue reward metric is reflecting well the combined results from dialogue efficiency and dialogue accuracy.

User satisfaction: Users evaluated the semi-learnt behaviour as the best. Although, semi-learnt behaviour was significantly different to fully-learnt behaviour, it was not significantly different to its deterministic counterpart. A similar user satisfaction result was found by Singh et al. (2002) and Lemon et al. (2006a). The performance of optimized confirmation strategies may be obscured by high recognition error rates. Future experiments could investigate optimized confirmation strategies under lower recognition error rates. In addition, the differences between learnt behaviours were statistically significant in the following qualitative metrics: *system understood*, *task easy*, *expected behaviour*, and *future use*. Similar differences were observed when comparing statistical significance between deterministic and fully learnt behaviour. These results suggest that those are the metrics with more impact on perceived system performance in the presence of unexpected dialogue behaviour such as infinite loops.

The results above can be summarized as follows (see also box plots of Figure 7.2). *First*, dialogues by deterministic and semi-learnt behaviour were more successful than dialogues by fully-learnt behaviour. These unsuccessful dialogues were reflected in the efficiency metrics, where fully-learnt behaviour falsely seems to be most efficient. *Second*, deterministic and semi-learnt behaviours are equally successful but the latter is more efficient. *Third*, real users perceived fully-learnt behaviour as the worst, and the other behaviours with equivalent medians. *Finally*, the problem of infinite loops could have been avoided (as in equation 5.11); however, if a spoken dialogue policy uses fully-learnt behaviour without a good reward function or without constraints to generate dialogues that make sense to humans, then it may not learn **successful and coherent behaviours**. According to the quantitative and qualitative results above, it can be concluded that semi-learnt behaviour was better than the other behaviours.

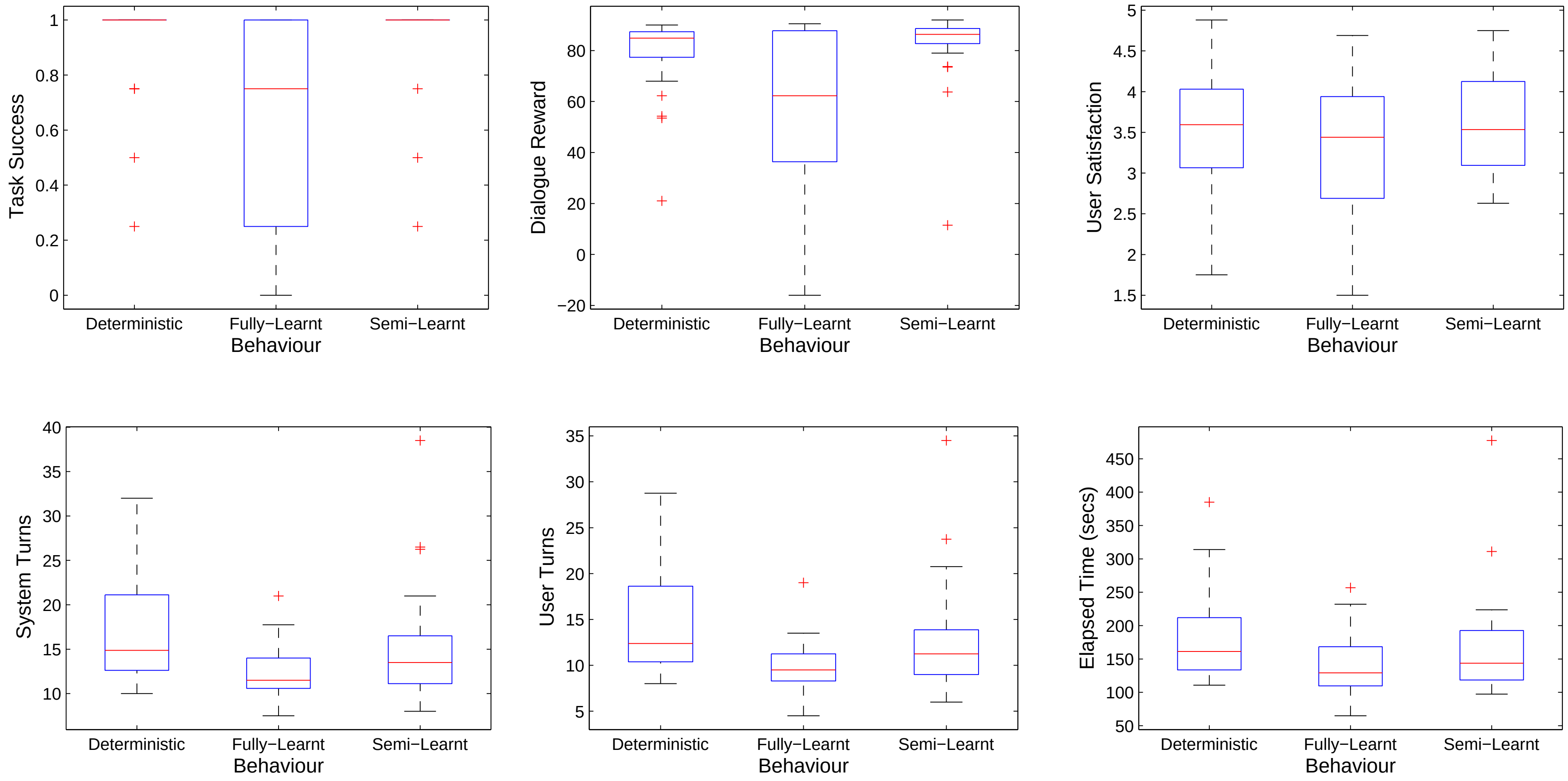


Figure 7.2: *Box plots of dialogue evaluation metrics per machine behaviour in the CSTR travel planning spoken dialogue system. The system performance in the top plots is interpreted as 'the higher the better' and in the bottom plots as 'the lower the better'.*

7.3.3.2 Analysis of results based on users with only successful dialogues

A further (and possibly more fair) comparison of behaviours was based on users with only successful dialogues[4] – shown in Table 7.6. It shows a summary of results comparing deterministic and fully-learnt behaviour against semi-learnt behaviour; including statistical significance. Firstly, it can be observed that both learnt behaviours were more efficient than their deterministic counterpart (in system/user turns, at $p \leq 0.05$), and the differences between learnt behaviours were not significant. Secondly, no significant differences were observed in dialogue quality. However, the statistical trend in event error rate suggests that the semi-learnt behaviour handled the trade-offs of acceptance /confirmation/rejection events more effectively. Thirdly, it can be noted that both learnt behaviours obtained more reward than their deterministic counterpart, and that therefore this metric is reflecting the significant differences observed from efficiency metrics. Last, similar to the results for all dialogues, the semi-learnt behaviour obtained the highest score in user satisfaction, but the differences were not significant.

These results confirm that semi-learnt dialogue behaviour is a better alternative than deterministic, and indicate that its performance is comparable to that of fully-learnt behaviour when they are evaluated on only successful dialogues.

Table 7.6: *Results of the CSTR travel planning spoken dialogue system using data from users – with only successful dialogues. They are organized in the following groups of metrics: dialogue efficiency, dialogue quality, task success and user satisfaction.*

Measure	Deterministic Behaviour$^{(1)}$	Fully-Learnt Behaviour$^{(2)}$	Semi-Learnt Behaviour$^{(3)}$	*p*-values		
				(1,2)	(1,3)	(2,3)
Avg. System Turns	14.58	11.94	12.58	≤ 0.05	≤ 0.05	
Avg. User Turns	12.50	9.75	10.23	≤ 0.05	≤ 0.05	
Avg. Time (secs)	159.74	142.69	132.48	≤ 0.05		
Word Error Rate	0.343	0.265	0.276			
Keyword Error Rate	0.209	0.137	0.167	≤ 0.1		
Event Error Rate	0.365	0.233	0.175		≤ 0.1	≤ 0.1
Avg. Task Success	1.00	1.00	1.00			
Avg. Dialogue Reward	85.42	88.06	87.42	≤ 0.05	≤ 0.05	
User Satisfaction	31.28	31.78	32.39			

[4]Users with only successful dialogues: 9 users out of 32, where each user did six dialogue tasks.

7.3.4 Evaluation of simulated behaviours

This section describes a quantitative analysis of simulated and real dialogue behaviours. For such a purpose, the performances of speech recognition, user behaviour, and machine behaviour were compared using the evaluation metrics of section 4.4.

7.3.4.1 Real versus simulated speech recognition

The real conversational environment used the ATK/HTK speech recognizer, and the simulated one used a simulated speech recognition error model (see section 4.3). Recognition results in terms of Keyword Error Rate (KER) for both environments were as follows: 20% in the simulated environment and 29% in the real one. For confidence scoring, the real environment showed confidence scores based on the probability density functions shown in Figure 7.3 (estimated from real data based on a normal density function), and the simulated environment generated uniformly distributed random confidence scores resulting in equal numbers of confidence levels. It can be observed that simulation used a more conservative KER and different distributions of confidence levels. This is because no training data was assumed, where the realistic probability distributions for recognition errors and confidence scoring were unknown.

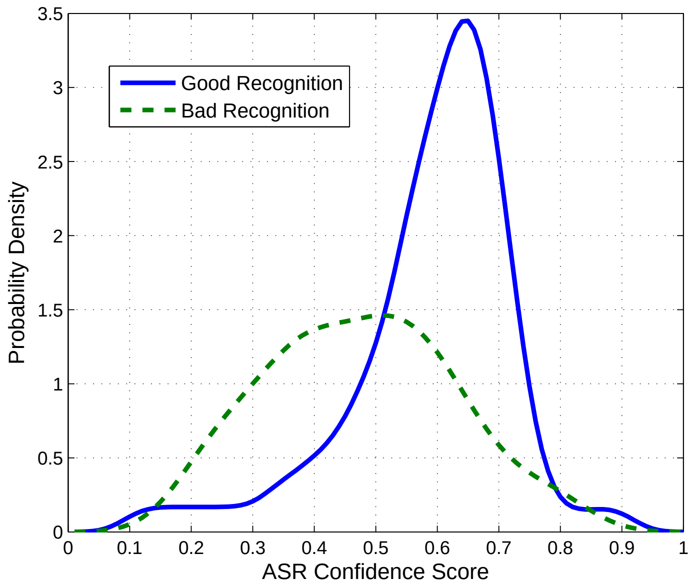


Figure 7.3: *Probability density functions estimated from observed speech recognition confidence scores of keywords in data collected by the CSTR travel planning system.*

Previous work in Automatic Speech Recognition (ASR) simulation has assumed that exponential probability distributions can model the behaviour of ASR confidence scorers (Pietquin, 2004; Williams, 2006). This research found that this assumption does not hold for the ASR system used here. Instead, the **gamma probability distributions** are suggested to simulate ASR confidence scores, which are more flexible and include the exponential distribution. Thus, learnt dialogue policies in a second stage can be retrained with more realistic ASR behaviour in order to generate potentially even better policies. Nevertheless, it was found that even conservative ASR error modelling was sufficient to find better dialogue policies than deterministic behaviour.

7.3.4.2 Real versus simulated user behaviour

Simulated user behaviour was compared against real user behaviour and against random user behaviour (see 2.4 for a review on dialogue simulation). For such a purpose three evaluation metrics were used: Precision-Recall based on the F-Measure score, dialogue similarity based on the Kulback-Leibler (KL) divergence, and Coherence Error Rate (CER). They were applied following the descriptions of section 4.5. The objectives of this evaluation were: (a) to observe if the simulated user model used to learn the dialogue strategies was a reasonable thing to use, and (b) to validate that dialogue realism could be distinguished by the proposed metrics (KL-divergence and CER).

This evaluation used three sets of user responses: (1) real user responses were extracted from annotated data from the realistic environment, consisting in 192 dialogues including 4623 user utterances; (2) simulated coherent responses used algorithm 5 described in section 4.3.1; and (3) simulated random responses used the same algorithm, but user dialogue acts were chosen randomly (at line 12) and with a random sequence of slots. It must be noted that all user responses (real, simulated coherent or simulated random) were derived from machine dialogue acts in the real logged data, which allows a more fair comparison. In addition, all user responses were not distorted because they were compared before speech recognition occurred.

Table 7.7 shows results of simulated user behaviour for two evaluation metrics: Precision-Recall and KL-divergence. It can be seen that both metrics agreed in the ranking of dialogue realism, including the proposed KL-divergence metric.

These results show that simulated coherent behaviour is more similar to real user behaviour than simulated random behaviour. It can be observed that the Precision-Recall of simulated coherent behaviour obtained higher scores than those reported before (Schatzmann et al., 2005b; Georgila et al., 2006), approaching the upper-bound scores from real user behaviour. To further analyze precision-recall results, the average of the more strict precision-recall ‘F-Measure’ was computed incrementally according

Table 7.7: *Evaluation of real and simulated user behaviour with Precision-Recall in terms of F-Measure (the higher the better) and KL-divergence (the lower the better).*

Compared Dialogues	F-Measure		KL-divergence
	less strict	more strict	
Real1 vs Real2	0.915	0.749	1.386
Real vs Simulated Coherent	0.708	0.612	4.281
Real vs Simulated Random	0.633	0.360	5.025
Simulated Coherent vs Simulated Random	0.417	0.247	6.532

Notes: (1) The less strict F-Measure score considers a user response as a sequence of actions, and the more strict score considers a user response as a single action, (2) the real dialogues were divided into two subsets ('Real1' and 'Real2') to provide an upper-bound score, (3) KL-divergence used Witten-Bell discounting to smooth the probability distributions.

to the size of the dialogue data. This is shown in Figure 7.4. It can be observed that the more real dialogue data the higher the precision-recall. This is because precision-recall is strictly penalizing unseen behaviour, and as more real data is observed, more varied user responses per machine action are possible to match simulated responses.

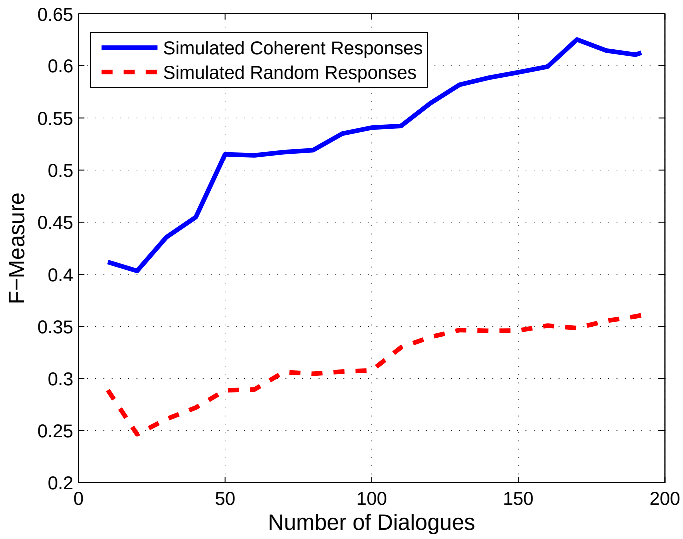


Figure 7.4: *F-measures of real vs. simulated user responses in function of the data size, showing that the more real dialogue data is used, the higher the precision-recall.*

In addition, the results in terms of Coherence Error Rate (CER) for real, simulated and random responses were 8.23%, 2.99%, 30.10%, respectively. The user responses with silences or incomplete dialogue acts were considered as incoherences because whatever the user said (e.g. partial words, out-of-vocabulary words, mumbles, etc.), no dialogue act could be extracted from the given utterance. It can be observed that simulated coherent behaviour behaved very optimistically, that is not very different from real user behaviour, and it is significantly different from the coherence of random behaviour. This metric is interesting because it evaluates a different perspective from the existing metrics, it may be used as a complementary evaluation, and future work may apply it to different data sets and domains to evaluate its significance.

7.3.4.3 Evaluating the baseline of machine dialogue behaviour

To evaluate the deterministic (hand-crafted) machine dialogue behaviour of the CSTR travel planning spoken dialogue system, the evaluation metric called 'Event Error Rate (EvER)' was used, defined by equation 4.13. For such a purpose, different confirmation strategies were proposed in Table 4.13, aiming to find a reasonable baseline of machine dialogue behaviour. The assumption here was that the confirmation strategy with the lowest EvER would be the best baseline. Real data (all keywords with their corresponding confidence scores) collected from the CSTR travel planning system was used to compute EvER for such confirmation strategies, see Table 7.8. It can be seen that the deterministic behaviour of choice in this research (Strategy3) indeed obtained the lowest EvER, together with 'Strategy4'. Although they obtained the same result, the former is more attractive, due to its use of implicit confirmations because it leads towards more efficient conversations. Therefore, it can be concluded that the learnt dialogue strategies used in the CSTR travel planning dialogue system were compared against a reasonable baseline of deterministic machine dialogue behaviour.

Table 7.8: *Event Error Rate (EvER) results of real dialogues for confirmation strategies of Table 4.13. Abbreviations: ca=correct acceptance, cc=correct confirmation, cr=correct rejection, fa=false acceptance, fc=false confirmation, fr=false rejection.*

Strategy	ca(%)	cc(%)	cr(%)	fa(%)	fc(%)	fr(%)	EvER(%)
Strategy1	73.6	0	0	26.3	0	0	26.3
Strategy2	71.9	2.2	0	17.0	9.3	0	26.3
Strategy3	26.7	44.6	9.3	2.5	14.4	2.2	19.2
Strategy4	0	71.4	9.3	0	17.0	2.2	19.2
Strategy5	0	73.6	0	0	26.3	0	26.3

7.3.5 Do people want to talk to spoken dialogue systems?

During the experiments with the CSTR travel planning spoken dialogue system – at the end of each participant session, participants were asked the following question: 'Would you use spoken dialogue systems for other tasks based on this experience?' Participants ranked their preference using a 5-point Likert scale, where the higher the score, the better the satisfaction. Figure 7.5 shows the results from this question, which is a combination of dialogue reward and preference for future use. It was noted that only 12%(4) percent of participants were pessimistic in their future use, 56%(18) of participants preferred to stay neutral, and 31%(10) were optimistic in its future use. The scores in preference of future use per user type were 3.0 for novice users, 3.28 for experienced users, and 3.2 for expert users (see p. 146 for proportions of user types). To further analyze this, consider splitting the group of participants: the first group with dialogue reward smaller than 80 and the rest in the second group. There was a 2.8 score in preference of future use for the first group of participants against a 3.7 score for the second group. Based on this result (significant at $p = 0.006$) it can be inferred that *the higher the dialogue reward the higher the preference for future use of dialogue systems*. This result can be related to the fact that dialogue strategies need high overall dialogue rewards to gain wider acceptance by real users. This should motivate the speech and language processing community to build more sophisticated spoken dialogue systems.

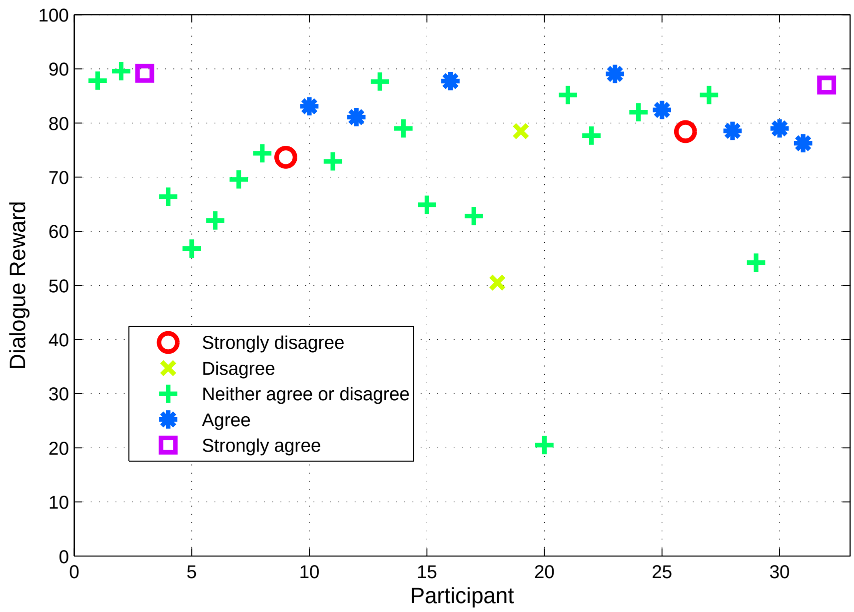


Figure 7.5: *Scatter plot showing participants' preference given the following question: 'Would you use spoken dialogue systems for other tasks based on this experience?'.*

7.4 Discussion and future directions

This section discusses the following issues derived from the experimental results described above: (1) coherent learnt dialogue behaviour, (2) unrealistic error simulation, and (3) robust semantic knowledge updates.

Firstly, a danger of learnt dialogue strategies is that they may yield incoherent behaviour, such as the fully-learnt behaviour reported in this thesis. This situation may happen if the reward function does not penalize bad actions properly. The importance of this issue increases as the dialogue system becomes larger, with more complex behaviours, where the avoidance of incoherent actions in fully-learnt behaviours is not guaranteed. Therefore, this research suggests that **spoken dialogue strategies should not only be optimal according to some performance measure, but also coherent in their action**s. The semi-learnt behaviour evaluated in this chapter ensured coherent behaviour through the use of partially specified dialogue strategies.

Secondly, the simulated conversational environment that was used did not model errors as in a real environment, which was to be expected due to the lack of training data. Nonetheless, the experimental results provided evidence to conclude that this heuristic-based dialogue simulation approach was useful for learning dialogue strategies with superior performance compared with a reasonable baseline of deterministic behaviour. This result is relevant for spoken dialogue systems in new domains, where annotated dialogue data is not available. The simulated environment could be enhanced with probability distributions estimated from real annotated data as in Schatzmann et al. (2007b). However, due to the fact that collecting training data is costly and time consuming, a potential for further research is to investigate methods for generalizing simulated behaviours for spoken dialogue systems across different domains.

Thirdly, one of the most important limitations of this work was the lack of a robust approach for updating slot values. Due to the fact that speech recognition hypotheses may include errors, it was difficult to know when to update or reject the recognised slot values. The effect of non-robust keyword updating is that the system eventually gives the impression of forgetting what has been said before. This highlights the importance of effective and efficient mechanisms for dialogue history tracking. Future research can incorporate beliefs into the knowledge-rich states of the proposed framework with ideas from approaches such as regression methods (Bohus and Rudnicky, 2005a), POMDPs (Roy et al., 2000; Williams, 2006), or Bayesian models (Horvitz and Paek, 1999, 2000; Paek and Horvitz, 2000; Williams, 2007d; Thomson et al., 2008).

7.5 Conclusions

A spoken dialogue system was presented using hierarchical reinforcement learning under the formalism of Semi-Markov decision processes, and its performance was investigated for three different types of machine dialogue behaviour: deterministic, fully-learnt and semi-learnt.

Semi-learnt behaviour was quantitatively better than the other dialogue behaviours. It achieved similar task success to deterministic behaviour ($\sim$ 95%) and more efficient conversations by using 9% fewer system turns, 12% fewer user turns, and 7% less time. It also outperformed fully-learnt behaviour by 35% in terms of higher task success. However, although fully-learnt behaviour resulted in inferior overall performance, it cannot be discarded as a better alternative than hand-crafted behaviour. But it is less flexible and less coherent than semi-learnt behaviour because it does not include a mechanism to guarantee coherent actions, which is essential for successful dialogues. On the other hand, whilst users did perceive significant qualitative differences between fully-learnt behaviour and the other behaviours, they did not observe significant differences between deterministic and semi-learnt behaviours.

The key findings in this chapter can be summarized as follows:

(1) hierarchical semi-learnt dialogue agents are a better alternative (with higher overall performance) than deterministic or fully-learnt behaviour;

(2) highly-coherent user behaviour and conservative recognition error rates (keyword error rate of 20%) were sufficient for learning dialogue policies with superior performance to a reasonable hand-crafted behaviour;

(3) learnt dialogue agents should include a mechanism to guarantee coherent behaviour;

(4) hierarchical reinforcement learning dialogue agents are feasible and promising for the (semi-) automatic design of optimized behaviours in larger-scale spoken dialogue systems.

Chapter 8

Conclusions and future work

This thesis investigated how to optimize the behaviour of information-seeking spoken dialogue systems in a scalable and efficient way under the reinforcement learning paradigm. It proposed two approaches for learning hierarchical dialogue strategies based on the Semi-Markov Decision Process (SMDP) model. The first approach used a hierarchy of SMDPs that ignore irrelevant state variables and actions, where the root SMDP represents the entire dialogue session and its child SMDPs represent subdialogues, and each child can have more descendants and so on, forming a hierarchy of SMDPs. The second approach extends the previous one by including partially specified dialogue strategies to learn only where necessary, providing the actions available per state for the current SMDP at runtime. It includes the HAM+HSMQ-Learning algorithm to find a hierarchy of optimal context-independent policies. In addition, this thesis proposed a heuristic dialogue simulation framework so that the reinforcement learning agents could acquire their behaviour automatically. In contrast to other dialogue strategy learning approaches, this research suggested learning a hierarchy of dialogue policies instead of a single one, simultaneously integrating hand-coded and learnt behaviours into a single framework. Experimental results in simulated and real environments provided evidence to conclude that both approaches scale well, and that hierarchical reinforcement learning agents are feasible and promising for the (semi) automatic design of adaptive behaviours in larger-scale dialogue systems. However, the second approach is more appealing with respect to dialogue as it outperforms hand-coded behaviour, and is more suitable for online learning in real environments.

The main contributions made by this thesis are: (1) the Semi-Markov Decision Process (SMDP) model for spoken dialogue; (2) the concept of partially specified dialogue strategies; and (3) the evaluation of learnt dialogue behaviours with real users.

8.1 Future work

This research suggests the following promising research avenues for endowing spoken dialogue systems with optimized, adaptive, robust and scalable behaviours.

8.1.1 Hierarchical dialogue action under uncertainty

The spoken dialogue system investigated here used the first-best recognition and understanding hypotheses. It is well known that such hypotheses are prone to errors. An important enhancement consists of keeping track of uncertain events such as recognised words, current dialogue goal, and type of user. This suggests that system beliefs need to be modelled at different levels of granularity. There are at least two approaches that can be investigated for such a purpose. First, POMDP-based dialogue approaches (Williams, 2006; Young et al., 2007; Henderson and Lemon, 2008) can be extended with a hierarchical setting (Theocarous, 2002; Theocarous et al., 2004; Pineau, 2004). Second, the approaches proposed in this thesis can be extended with an additional probabilistic knowledge base (e.g. belief network) to maintain dialogue information under uncertainty. This would help to balance the issues of robustness and scalability into an integrated framework.

8.1.2 Learning more complex dialogue strategies

This thesis focused on optimizing confirmation strategies to keep their assessment simple rather than evaluating multiple dimensions. Nonetheless, there is a wide range of optimized dialogue behaviours that can be incorporated into this kind of system. For example: learning initiative strategies (Litman et al., 2000; Walker, 2000), learning to give help (Frampton and Lemon, 2006), learning to ground (Pietquin, 2007), learning to present information (Rieser and Lemon, 2007), learning to clarify (Rieser and Lemon, 2006a), learning to negotiate (English and Heeman, 2005), learning to recover from errors (Bohus, 2007; Skantze, 2007; Frampton and Lemon, 2008), learning multimodal strategies (Rieser and Lemon, 2008), and learning to collaborate. The thorough integration of all these behaviours into a single framework remains to be investigated. This would require the support of learning on large search spaces – hence the importance of this topic. The underpinning ideas of the proposed approaches are appealing for such a purpose. In general, the long-term goal is to build spoken dialogue systems with behaviours that approximate better to more natural conversations.

8.1.3 Learning reusable dialogue strategies

The proposed reinforcement learning algorithm and many other algorithms in the literature update values for each individual state-action pair. It would be useful if they could apply such updates in more than one situation. The hierarchical nature of the proposed approaches allows the reuse of complete dialogue policies, but the reuse of similar behaviours remains to be investigated. Several approaches have been proposed by machine learning researchers and they could be applied to spoken dialogue (Konidaris and Barto, 2007; Asadi and Huber, 2007; Wilson et al., 2007; Taylor and Stone, 2007). This is also known in the literature of reinforcement learning as 'knowledge transfer'. Methods with such capacity would increase the learning speed, and facilitate the deployment of spoken dialogue systems with reusable dialogue behaviours.

8.1.4 Hierarchical dialogue control using function approximation

The proposed approaches include support for tabular hierarchical reinforcement learning. However, if a given subtask is intractable (i.e. the state-action space becomes too large and indecomposable) then alternative methods should be adopted to make such subtasks feasible. One of the most promising approaches reported in the literature of reinforcement learning is that of function approximation. The approaches proposed in this thesis could be combined with function approximators such as neural networks or linear function approximation (Henderson et al., 2005). Furthermore, this research avenue opens the possibility of learning spoken dialogue behaviours combining (sub) solutions derived from different reinforcement learning approaches.

8.1.5 Safe dialogue state abstraction

In the proposed approaches the system designer has manually to remove irrelevant state variables and actions for each subtask. This was essential for dramatically reducing the state-action space. Although this is useful because it allows the system designer to specify what to remove, it may become problematic if relevant information is removed, leading to unsafe state abstraction. Therefore, it would be useful to have a method for performing state abstraction of dialogue information in a safer way (Dietterich, 1999; Andre and Russell, 2002; Jong and Stone, 2005). In addition, previous work on dialogue-based feature selection can be extended with a hierarchical setting.

8.1.6 Hierarchy discovery of dialogue subtasks

In the proposed approaches the system designer has to specify the hierarchy of subtasks manually. Although specifying hierarchies may be intuitive – such as writing the structure of an object oriented program, it would be useful if the dialogue hierarchy could be inferred from data or interactions with an environment. Such methods might allow the finding of better hierarchies than the manually designed ones, although so far they have been investigated only in small-scale navigation domains (McGovern, 2002; Hengst, 2003). The previous topic would offer useful results for such a purpose.

8.1.7 Hierarchical dialogue reward functions

The current practice of reinforcement learning for spoken dialogue uses a single reward function. Although the proposed approaches in this thesis allowed the use of a different reward function per subtask, the experimental setting used the same performance function across the entire hierarchy. Intuitively, hierarchical dialogue optimizations such as those described in subsection 8.1.2 may require different types of reward function at different levels of granularity. Moreover, as the dialogue complexity increases, it becomes more difficult to specify such performance functions. It remains to be investigated how to specify or infer such hierarchical reward functions once dialogue data has been collected and annotated. The PARADISE evaluation framework may be explored for this purpose (Walker, 2000).

8.1.8 Online dialogue strategy learning from real users

Currently available approaches for dialogue strategy learning – including the proposed ones – learn behaviour in an offline fashion. This means that learnt behaviours are derived either from simulated conversational environments or from collected dialogues. An alternative approach is to (re) learn online from real human-machine interactions. This research direction applied to large-scale systems would require very efficient learning methods: the issue of coherent dialogue behaviour becomes crucial, moreover several of the previously proposed research avenues might help for such a purpose.

8.1.9 Task-independent dialogue simulation

The proposed approaches used a heuristic model for simulating human-machine dialogues. Alternative approaches train probabilistic simulation models from dialogue data. However, every time a new spoken dialogue system is built, a new dialogue simulator is required. A more practical approach would be to have a generic dialogue simulator that can be used in systems for different domains. Such kinds of simulator should understand a wide range of behaviours with a common notation across dialogue systems. Even if the previous research avenue becomes feasible, such simulators would be useful for deploying behaviours with an initial optimization.

8.1.10 Richer knowledge representations

The knowledge representation in the proposed approaches is rudimentary and so limits the expressive description of complex situations and actions, and may be more appropriate for other types of interaction such as negotiation or collaborative dialogues in human-robot interaction. The emerging field of relational reinforcement learning (Dzeroski et al., 2001; Tadepalli et al., 2004) and hybrid approaches (Ryan, 2002) might be investigated. Alternatively, the knowledge base of the proposed approaches could be augmented not only with belief networks but also with hierarchical relational structures. This work could be based on an integrated knowledge base for robust and adaptive dialogue strategy learning of more complex conversations. In general, dialogue knowledge representation is an important research topic for endowing reinforcement learning spoken dialogue agents with robust and descriptive knowledge.

8.1.11 A benchmark framework for spoken dialogue strategies

It is well known that the progress of spoken dialogue strategies is difficult to assess. The lack of standards makes the comparison of new spoken dialogue strategies against state-of-the-art ones difficult. Several computer science communities evaluate their methods or agents on standardized software frameworks or resources. For instance, the reinforcement learning community organizes the ‘Reinforcement Learning Competition’[1] to compare the performance of their methods. The robotics community organizes the ‘RoboCup Soccer Competition’[2] to compare their methods embedded into robots. The speech synthesis community organizes the ‘Blizzard Challenge’[3] to compare their techniques. Such kinds of initiative would be very valuable in assessing progress for spoken dialogue research.

[1] http://rl-competition.org/
[2] http://www.robocup.org/
[3] http://festvox.org/blizzard/

8.2 Findings

The following findings were derived from this research:

(i) **Hierarchical task decomposition with state-action abstraction reduces search spaces dramatically** Although this is not new, it confirms the claim that top-down hierarchical control reduces the complexity of decision makers from exponential to linear in the size of the problem. Experimental results in flight-booking and travel planning systems report state-action space reductions of more than 99%. This highlights the importance of this approach for large-scale systems.

(ii) **Hierarchical reinforcement learners find solutions faster than flat learners** This is derived from learning dialogue behaviours on reduced state-action spaces rather than full ones. Experiments on a simulated spoken dialogue system in the flight-booking domain reported that hierarchical reinforcement learning converged roughly four orders of magnitude faster than flat reinforcement learning.

(iii) **Hierarchical reinforcement learning agents find near-optimal solutions** This is not new either, but confirms the claim by machine learning researchers that hierarchical reinforcement learners may find solutions with slight sub-optimalities. Experiments on a flight-booking system report a small loss in optimality of 0.3 more system turns than flat learning, resulting in slightly longer dialogues.

(iv) **Hierarchical learnt dialogue strategies can outperform reasonable hand-coded baselines** Experimental results report that hierarchical learnt dialogue strategies are better than a reasonable hand-coded behaviour (this baseline outperformed other hand-coded dialogue behaviours on real data). This was found in both simulated and real conversational environments. However, the benefits in the real environment were smaller than its counterpart due to the use of a simpler simulated dialogue model for dialogue strategy learning.

(v) **Semi-learnt dialogue policies are a good alternative to fully-learnt or deterministic behaviour** Experimental results report the propensity of fully-learnt behaviours to learn incoherent actions, possibly due to the fact that reward functions do not penalize bad actions correctly. This problem is reduced in semi-learnt behaviours because by learning only where necessary; even if they do not explore the search space completely they will take coherent actions. Semi-learnt behaviours are also appealing because they can find the best actions (according to reward functions) that might not be easy to specify for a system designer.

(vi) **Real users act with highly coherent behaviour at the dialogue act level** The experiments reported in this thesis reveal that real users in task-oriented conversations behaved coherently 92% of the time. This result rated the incoherent user dialogue acts against all user dialogue acts, and can be taken into account in simulating user behaviour.

(vii) **Fully-coherent user behaviour and conservative recognition error rates are sufficient for learning better policies than hand-coded behaviour** The simulated conversational environment employed in this research used fully-coherent user behaviour and distorted user dialogue acts with 20% of recognition error rates with a flat distribution. This setup was sufficient to learn a spoken dialogue behaviour that was more efficient than a deterministic one.

(viii) **Learnt dialogue policies should include a mechanism to guarantee coherent behaviour** Experimental results report that fully-learnt behaviour may not learn the best actions per state, and possibly behave incoherently when testing the learnt policy. This may be due to the following situations: (1) simple reward functions; (2) insufficient exploration during learning; and (3) incorrect state transitions. Experimental results confirm that the first situation (and potentially the second as well) can be avoided by constraining the actions available to only situation-action pairs that make sense to humans.

(ix) **The proposed approaches can be applied to larger-scale dialogue systems** This research implemented a real spoken dialogue system in the travel planning domain with five dialogue goals and 26 slots of information. This is the largest scale spoken dialogue system so far (in terms of dialogue goals and slots) tested using the reinforcement learning paradigm. Although it focused on optimizing confirmation strategies, the proposed framework supports larger-scale systems with a wider range of optimized behaviours, which is essential to build more sophisticated conversational agents.

Appendix A

Notation

Table A.1: *Notation for human-machine dialogue modelling.*

Symbol	Description
k_t^m	Machine's knowledge base at time t
k_t^u	Simulated user's knowledge base at time t
π^m	Machine's dialogue strategy
π^u	Simulated user's dialogue strategy
s_t^m	Machine dialogue state at time t
w_t^m	Joint machine dialogue state at time t
s_t^u	User dialogue state at time t
a_t^m	Machine dialogue act at time t
a_t^u	User dialogue act at time t
$\tilde{a}^m$	Distorted machine dialogue act at time t
$\tilde{a}^u$	Distorted user dialogue act at time t
x_t^m	Machine speech signals at time t
x_t^u	User speech signals at time t
w_t^m	Machine words at time t
w_t^u	User words at time t
c_t^m	Machine keywords at time t
c_t^u	User keywords at time t
$(s_t^m, a_t^m, s_t^u, a_t^u)$	User-machine interaction at the dialogue act level
(s_t^m, D_j^i)	Sub-dialogue of user-machine interactions in state s_t^m
D	Dialogue of user-machine interactions

Table A.2: *Notation for flat and hierarchical reinforcement learning.*

Symbol	Description
t	Discrete time step
T	Final time step
s_t	State at time t
a_t	Action at time t
r_t	Reward at time t
π	Policy
$\pi(s)$	Action taken in state s
S	Set of environment states
$A(s)$	Set of all possible actions in state s
$P(s'\|s,a)$	Probability of transition from s to s' under action a
$R(s'\|s,a)$	Expected reward for taking action a in s transitioning to s'
$V^{\pi}(s)$	Value of state s under policy π
$V^{*}(s)$	Value of state s under optimal policy π^{*}
$Q^{\pi}(s,a)$	Value of taking action a in state s under policy π
$Q^{*}(s,a)$	Value of taking action a in state s under optimal policy π^{*}
γ	Discount rate parameter
α	Step size parameter
τ	Discrete multiple time-step
$\bar{s}_n$	Abstract machine state at time n
$\mathcal{M} = \{M_0^0, \ldots, M_j^i\}$	Hierarchy of Semi-Markov Decision Processes (model j at level i)
$\mathcal{H} = \{H_0^0, \ldots, H_j^i\}$	Hierarchy of abstract machines (HAM j at level i)
$\mathcal{M}' = \{M_0'^0, \ldots, M_j'^j\}$	Hierarchy of induced Semi-Markov Decision Processes (SMDPs)
$M_j'^i =< S_j'^i, A_j'^i, T_j'^i, R_j'^i >$	Induced Semi-Markov Decision Processes (model j at level i)
$\pi = \{\pi_0^0, \ldots, \pi_j^i\}$	Hierarchical policy
π_j^i	Policy for SMDP j at level i
π_j^{*i}	Optimal policy for SMDP j at level i
S_j^i	Set of environment states for SMDP M_j^i
A_j^i	Set of actions for SMDP M_j^i
$P^{\pi_j^i}(s',\tau\|s,a)$	Probability of transition from s to s' under a lasting τ time steps
$R^{\pi_j^i}(s',\tau\|s,a)$	Expected cumulative reward for taking action a in s transitioning to s'
$V_j^{*i}(s)$	Value of state s under optimal policy π_j^{*i}
$Q_j^{*i}(s,a)$	Value of taking action a in state s under optimal policy π_j^{*i}
γ^{τ}	Discount rate for executing action a lasting τ time steps

Appendix B

Dialogue data structures

The dialogue data structures described in this appendix have the purpose of representing knowledge about the conversation for the simulated user and machine. They are referred to the human-machine dialogue simulation framework described in chapter 4. The data structures are briefly described as follows.

- Table B.1 shows the classes used to build the knowledge base of the simulated user. They are instantiated or re-initialized for each simulated conversation, and were implemented with hash tables for fast information retrieval. These classes are only used during simulation; on real conversations they are ignored. Section 4.3.1 explains how to use them.

- Table B.2 shows the classes used to build the machine's knowledge base. They are instantiated or re-initialized for each real or simulated conversation, and were also implemented with hash tables. See also section 4.3.1 for how to use them. These classes only included the first hypotheses of recognition and parsing events; however, they can be updated from an additional probabilistic knowledge base to mitigate uncertainty in the conversation.

- Tables B.3 and B.4 are used to generate the state-action space of the flight booking dialogue system. This state representation only includes state variables for flat dialogue optimization. Table 5.1 extends this set of state variables for hierarchical dialogue optimization.

- Tables B.5, B.6, and B.7 are used to generate the state-action space of the travel planning dialogue system. This state representation also includes state variables for flat dialogue optimization. Table 5.2 extends this set of state variables for dialogue optimization with a hierarchical setting.

Table B.1: *Description of dialogue-based classes to represent user knowledge.*

Class	Attribute	Values
DialogueFocus	lastUserDA	last user dialogue act
	lastMachineDA	last received machine dialogue act
	goalInFocus	current dialogue goal $g_i \in G$
	frameInFocus	current semantic frame $f_j \in F^{g_i}$
	slotInFocus	current information slot $c_k \in C^{g_i}_{f_j}$
DialogueAct	dialogueAct	dialogue act type with slot-value pairs
	dialogueActType	[*dialogue act types from table 4.1*]
	slotValues	a set of slot-value pairs
DialogueGoal	goalID	$g_i \in G = \{g_0, \ldots, g_{\|G\|-1}\}$
	goalStatus	{0=unfilled, 1=filled, 2=acknowledged, 3=relaxed}
	frames	a set of instances of the class *SemanticFrame*
SemanticFrame	frameID	$f_j \in F = \{f_0, \ldots, f_{\|F\|-1}\}$
	frameStatus	{0=unfilled, 1=filled, 2=confirmed, 3=relaxed}
	frameType	{non-terminal,terminal}
	acknowledged	{0=no, 1=yes}
	slots	a set of instances of the class *Slot*
Slot	slotID	$c_i \in C = \{c_0, \ldots, c_{\|C\|-1}\}$
	slotValue	keyword of the users's goal (e.g., flight/hotel/car)
	slotStatus	{0=unprovided, 1=provided, 2=reprovided 3=confirmed, 4=relaxed}
	retries	{0, 1, 2, 3}
	explicitConfirmations	{0, 1,...}
	implicitConfirmations	{0, 1,...}
Recognition	ker	keyword error rate, default= 0.1
	obedience	probability of providing slot in focus, default= 0.8
	multiSlotFilling	probability of providing other slots, default= 0.4
	negativeConfirmation	probability of saying "no" in explicit confirmations, without reproviding slots, default= 0.2

Table B.2: *Description of dialogue-based classes to represent machine knowledge.*

Class	Attribute	Values
DialogueStatus	salutation	{0=null, 1=greeted, 2=closed}
	completion	{0=non-started, 1=in-progress, 2=completed}
	topicShift	{0=none,1=pending}
	infoPresentation	{0=unprovided, 1=provided}
DialogueFocus	lastMachineDA	last machine dialogue act
	lastUserDA	last received user dialogue act
	goalInFocus	current dialogue goal $g_i \in G$
	frameInFocus	current semantic frame $f_j \in F^{g_i}$
	slotInFocus	current information slot $c_k \in C_{f_j}^{g_i}$
	grammarInFocus	current grammar $\in \{flights, hotels, cars, yesno\}$
DialogueAct	dialogueAct	dialogue act type with slot-value pairs
	dialogueActType	[*dialogue act types from table 4.1*]
	slotValues	a set of slot-value pairs
DialogueGoal	...	[*similarly as in table B.1*]
SemanticFrame	frameID	$f_j \in F = \{f_0, \ldots, f_{\|F\|-1}\}$
	frameStatus	{0=unfilled, 1=filled, 2=confirmed, 3=relaxed}
	frameType	{initial,mandatory,optional,terminal}
	acknowledged	{0=no, 1=yes}
	slots	a set of instances of the class *Slot*
Slot	slotID	$c_i \in C = \{c_0, \ldots, c_{\|C\|-1}\}$
	slotValue	keyword in the recognition dictionary
	confScore	speech recognition confidence score $[0, \ldots, 1]$
	slotStatus	{0=unfilled, 1=low confidence (conf.), 2=medium conf., 3=high conf., 4=confirmed}
	retries	{0, 1, 2, 3}
	explicitConfirmations	{0, 1,...}
	implicitConfirmations	{0, 1,...}
Recognition	ker	keyword error rate, default$= 0.2$
	lowConfidence	proportion of low confidence values, default$= 1/3$
	medConfidence	proportion of medium conf. values, default$= 1/3$
	highConfidence	proportion of high conf. values, default$= 1/3$
DatabaseInfo	dbQuery	SQL statement
	dbResult	{0=null, 1=none, 2=few, 3=many}
	dbTuples	retrieved database tuples

Table B.3: *State variables for the 6-slot flight booking spoken dialogue system.*

Variable	Values	Description
C00	$\{0,1,2,3,4\}$	Status of mandatory slot 'departure city'
C01	$\{0,1,2,3,4\}$	Status of mandatory slot 'destination city'
C02	$\{0,1,2,3,4\}$	Status of mandatory slot 'date'
C03	$\{0,1,2,3,4\}$	Status of mandatory slot 'time'
C04	$\{0,1,2,3,4\}$	Status of optional slot 'airline'
C05	$\{0,1,2,3,4\}$	Status of termianl slot 'flight offer'
SIF	$\{0,\ldots,5\}$	Slot in focus
DBT	$\{1,2,3\}$	Size of database tuples

Notes on domain values of state variables: C0?={0=unfilled, 1=low confidence, 2=medium confidence, 3=high confidence, 4=confirmed}; SIF={0=departure city, 1=destination city, 2=date, 3=time, 4=airline, 5=flight offer}; DBT={1=none, 2=few, 3=many}.

Table B.4: *Action space for the 6-slot flight booking spoken dialogue system.*

#	Action	Description
01	req	Request slot in focus
02	apo+req	Apology for mis-recognition + request slot in focus
03	sic+req	Single implicit confirmation + request slot in focus
04	mic+req	Multiple implicit confirmation + request slot in focus
05	sec	Single explicit confirmation of the slot in focus
06	mec	Multiple explicit confirmation of filled slots
07	acc	Move to the next ascending slot with lower-value (see example in the dialogue shown in page 35)
08	dbq+sta	Perform a database query + inform the database status
09	pre+ofr	Information presentation + offer options
10	apo+ofr	Apology for mis-recognition + offer options

Table B.5: *Dialogue goals in the 26-slot travel planning spoken dialogue system.*

Goal ID	Description
G00	Metagoal for flight booking (outbound and return flights)
G01	Requests, offers, and acknowledges information for a outbound flight
G02	Requests, offers, and acknowledges information for a return flight
G03	Requests, offers, and acknowledges information for a hotel room
G04	Requests, offers, and acknowledges information for a car
G05	Summarizes, offers and acknowledges information of flights, hotel and car

Table B.6: *Action space for the 26-slot travel planning spoken dialogue system.*

#	Action	Description
01	req	Request slot in focus
02	apo+req	Apology for mis-recognition + request slot in focus
03	sic+req	Single implicit confirmation + request slot in focus
04	mic+req	Multiple implicit confirmation + request slot in focus
05	sec	Single explicit confirmation of the slot in focus
06	mec	Multiple explicit confirmation of filled slots
07	acc	Move to the next ascending slot with lower-value
08	dbq+sta	Perform a database query + inform the database status
09	pre+ofr	Information presentation + offer options
10	apo+ofr	Apology for mis-recognition + offer options
11	ofr	Offer database options
12	rel	Relax slots of dialogue goal in focus
13	ack	Acknowledgement of dialogue goal in focus
14	gre	Greeting
15	clo	Good bye

Table B.7: *State variables for the 26-slot travel planning spoken dialogue system.*

Variable	Values	Description
SAL	$\{0,1,2\}$	Status of salutation: null, greeting, closing
GIF	$\{0,1,2,3,4,5\}$	Dialogue goal in focus
SGF	$\{0,1,2,3\}$	Status of goal in focus
C00	$\{0,1,2,3,4\}$	Status of mandatory slot ‘departure city’ of goal G01
C01	$\{0,1,2,3,4\}$	Status of mandatory slot ‘destination city’ of goal G01
C02	$\{0,1,2,3,4\}$	Status of mandatory slot ‘departure date’ of goal G01
C03	$\{0,1,2,3,4\}$	Status of mandatory slot ‘departure time’ of goal G01
C04	$\{0,1,2,3,4\}$	Status of mandatory slot ‘airline’ of goal G01
C05	$\{0,1,2,3,4\}$	Status of mandatory slot ‘flight type’ of goal G01
C06	$\{0,1,2,3,4\}$	Status of optional slot ‘airport’ of goal G01
C07	$\{0,1,2,3,4\}$	Status of terminal slot ‘choice’ of goal G01
C15	$\{0,1,2,3,4\}$	Status of mandatory slot ‘return date’ of goal G02
C16	$\{0,1,2,3,4\}$	Status of mandatory slot ‘return time’ of goal G02
C17	$\{0,1,2,3,4\}$	Status of terminal slot ‘choice’ of goal G02
C18	$\{0,1,2,3,4\}$	Status of initial slot ‘want hotel’ of goal G03
C19	$\{0,1,2,3,4\}$	Status of mandatory slot ‘location’ of goal G03
C20	$\{0,1,2,3,4\}$	Status of mandatory slot ‘price’ of goal G03
C21	$\{0,1,2,3,4\}$	Status of mandatory slot ‘brand’ of goal G03
C22	$\{0,1,2,3,4\}$	Status of terminal slot ‘choice’ of goal G03
C23	$\{0,1,2,3,4\}$	Status of initial slot ‘want car’ of goal G04
C24	$\{0,1,2,3,4\}$	Status of mandatory slot ‘cat type’ of goal G04
C25	$\{0,1,2,3,4\}$	Status of mandatory slot ‘location’ of goal G04
C26	$\{0,1,2,3,4\}$	Status of mandatory slot ‘pickup date’ of goal G04
C27	$\{0,1,2,3,4\}$	Status of mandatory slot ‘pickup time’ of goal G04
C28	$\{0,1,2,3,4\}$	Status of mandatory slot ‘rental days’ of goal G04
C29	$\{0,1,2,3,4\}$	Status of optional slot ‘rental company’ of goal G04
C30	$\{0,1,2,3,4\}$	Status of terminal slot ‘choice’ of goal G04
C31	$\{0,1,2,3,4\}$	Status of mandatory slot ‘want summary’ of goal G05
C32	$\{0,1,2,3,4\}$	Status of terminal slot ‘book trip’ of goal G05
SIF	$\{0,...,7,16,...,32\}$	Slot in focus
DBT	$\{0,1,2,3\}$	Number of database tuples of current goal
PRE	$\{0,1\}$	Status of information presentation in goal
ACK	$\{0,1\}$	Status of acknowledgement for current goal

Notes on domain values of state variables: GIF=$\{$0=flight booking, 1=outbound flight, 2=return flight, 3=hotel booking, 4=car rental, 5=summarize trip$\}$; SGF=$\{$0=unfilled, 1=filled, 2=confirmed, 3=relaxed$\}$; the domain values of variables for slots are the same as in TableB.3.

Appendix C

Sample hierarchical dialogue

This appendix describes a real dialogue between a user and the CSTR travel planning spoken dialogue system using a semi-learnt hierarchical reinforcement learning dialogue agent. This dialogue agent – optimized for efficient conversations – chooses hierarchical actions with a divide and conquer approach, according to knowledge in the hierarchical states[1]. The interested reader is referred to chapter 6 for more details about semi-learnt hierarchical dialogue control, to chapter 4 for information about dialogue acts, and to chapter 7 for information about the CSTR travel planning system.

Briefly, the hierarchical dialogue can be traced as follows: the machine is in the root subtask $M_0'^0$ and state *0,0,0,0,0,0*, and selects the primitive action *gre* corresponding to a greeting. Then in the same subtask the machine is in an updated state *0,1,0,0,0,0*, where it selects the composite action $M_0'^1$ (in charge of booking flights), by invoking this subtask the new dialogue state is *1,0,0*. Then it selects another composite action $M_0'^2$ (in charge of booking a single flight), and its initial state is *0,0,0,0,0*. In this subtask it selects the composite action $M_0'^3$ (in charge of collecting mandatory slots) and its initial state is *0,0,0,0,0,0,0*, here the strategy selects actions for collecting slots until finding the terminal state *4,4,4,4,4,4,4*. Once the subtask has been completed, control returns to its parent subtask $M_0'^2$ with an updated state *2,0,2,2,2*. Here the state was updated to ignore collecting optional slots and terminal slot. Then returns to its parent subtask $M_0'^1$ with an updated state *1,2,0*, from which it calls the subtask $M_1'^2$ (in charge of booking the return flight). The strategy is now in the initial state *0,0,0,0* and selects action $M_3'^3$ (in charge of collecting mandatory slots), once completing that subtask it returns control to its parent in order to select action $M_4'^3$, which completes the return flight booking; and so on until the end of the conversation.

[1]The choice states constrain the actions to be available in the environment states.

Table C.1: *Real conversation in the CSTR travel planning spoken dialogue system using semi-learnt hierarchical dialogue control (PART 1).*

Agent	Subtask	Env. State	Choice State	Action	Dialogue Act	Utterance
SYS	$M_0'^0$	0,0,0,0,0,0	choice1	gre	gre()	Welcome to the CSTR travel planning system.
	$M_0'^0$	0,1,0,0,0,0	choice2	$M_0'^1$		
	$M_0'^1$	1,0,0	choice4	$M_0'^2$		
	$M_0'^2$	0,0,0,0,0	choice5	$M_0'^3$		
	$M_0'^3$	0,0,0,0,0,0,0	choice8	req	req(C00)	Tell me your flight information.
USR					pro(C05=return,C00=edinburgh, C01=amsterdam,C02=january 22, C03=morning)	**I would like a return flight from Edinburgh to Amsterdam on january twenty second.**
ASR						*I would like a return flight from rome go to Amsterdam on january twenty second.*
SYS	$M_0'^3$	0,1,1,2,0,0,2	choice10	apo+req	apo()+req(C00)	Try again. Tell me your flight information. For example you can say: a return flight from Edinburgh to london leaving on the eighteenth of november in the afternoon travelling with British Airways.
USR					pro(C05=return,C00=edinburgh, C01=amsterdam,C02=january 22	**I a return flight from Edinburgh to Amsterdam leaving on january twenty second**
ASR						*I a return flight from Edinburgh to Amsterdam leaving on january twenty second*
SYS	$M_0'^3$	0,2,2,2,0,0,1	choice10	mec	mec(C05=return,C00=edinburgh, C01=amsterdam,C02=january 22)	Did you say a return flight from Edinburgh to Amsterdam leaving on january twenty second?
USR					con(C05=yes,C00=yes, C01=yes,C02=yes)	**Yes**

Table C.2: *Real conversation in the CSTR travel planning spoken dialogue system using semi-learnt hierarchical dialogue control (PART 2).*

Agent	Subtask	Env. State	Choice State	Action	Dialogue Act	Utterance
ASR						*Yes*
SYS	$M_0^{\prime 3}$	3,4,4,4,0,0,4	choice8	req	req(C03)	At what time do you prefer to travel?
USR					pro(C03=morning)	**In the morning**
ASR						*In the morning*
SYS	$M_0^{\prime 3}$	3,4,4,4,2,0,4	choice9	sec	sec(C03=morning)	Did you say a flight in the morning?
USR					con(C03=yes)	**Yes**
ASR						*Yes*
SYS	$M_0^{\prime 3}$	4,4,4,4,4,0,4	choice8	req	req(C04)	What is your preferred airline?
USR					pro(C04=british airways)	**British Airways**
ASR						*British Airways*
SYS	$M_0^{\prime 3}$	4,4,4,4,4,2,4	choice9	sec	sec(C04=british airways)	Did you say a flight travelling with British Airways?
USR					con(C04=yes)	**Yes**
ASR						*Yes*
SYS	$M_0^{\prime 3}$	4,4,4,4,4,4,4				[terminal state]
SYS	$M_0^{\prime 2}$	2,0,2,2,2				[terminal state]
SYS	$M_0^{\prime 1}$	1,2,0	choice4	$M_1^{\prime 2}$		
SYS	$M_1^{\prime 2}$	0,0,0,0	choice5	$M_3^{\prime 3}$		
SYS	$M_3^{\prime 3}$	15,0,0	choice8	req	req(C15)	When do you want to return?
USR					pro(C15=february 01)	**January no february first**
ASR						*January third in the morning Air France*

Table C.3: *Real conversation in the CSTR travel planning spoken dialogue system using semi-learnt hierarchical dialogue control (PART 3).*

Agent	Subtask	Env. State	Choice State	Action	Dialogue Act	Utterance
SYS	$M_3'^3$	15,2,1	choice10	mec	mec(C15=january 03, C16=morning)	Did you say a flight returning on january third In the morning?
USR					rep(C15=february 01)	**No a flight on february first**
ASR						*No i flying on february first*
SYS	$M_3'^3$	15,2,1	choice10	mec	mec(C15=february 01, C16=morning)	Did you say a flight returning on february first in the morning?
USR					pro(C16=evening)	**In the evening**
ASR						*In the evening*
SYS	$M_3'^3$	15,2,2	choice10	mec	mec(C15=february 01, C16=morning)	Did you say a flight returning on february first in the evening?
USR					con(C15=yes,C16=yes)	**Yes**
ASR						*Yes*
SYS	$M_3'^3$	15,4,4				[terminal state]
SYS	$M_1'^2$	0,0,2,0	choice5	dbq+sta		Please wait while I retrieve information.
SYS	$M_1'^2$	2,0,2,0	choice5	$M_4'^3$		
SYS	$M_4'^3$	0,0,0,0	choice14	pre+ofr	pre(db tuples)+ofr(C17)	I found 2 return flights from Edinburgh to Amsterdam with British Airways. Choice 1: departing at six fourty five a.m. and arriving at eleven twenty a.m. The return flight departs at four fourty five p.m. and arrives at seven fifty p.m. The price is 147 pounds and 60 pence. Choice 2: departing at ten a.m.

Table C.4: *Real conversation in the CSTR travel planning spoken dialogue system using semi-learnt hierarchical dialogue control (PART 4).*

Agent	Subtask	Env. State	Choice State	Action	Dialogue Act	Utterance
						and arriving at three five p.m. The return flight departs at four fourty five p.m. and arrives at seven fifty p.m. The price is 152 pounds and 60 pence. Which choice do you prefer?
USR					pro(C17=one)	**Choice one**
ASR						*Choice one*
SYS	$M_4'^3$	0,0,1,2	choice15	sec	sec(C17=one)	Did you say choice one?
USR					con(C17=yes)	**Yes**
ASR						*Yes*
SYS	$M_4'^3$	0,0,1,4	choice16	ack	ack(G02)	I have added this flight to your itinerary.
SYS	$M_4'^3$	1,0,1,4				[terminal state]
SYS	$M_1'^2$	2,0,2,2	choice5			[terminal state]
SYS	$M_0'^1$	2,2,2	choice4			[terminal state]
SYS	$M_0'^0$	3,1,2,0,0,0	choice2	$M_2'^2$		
SYS	$M_2'^2$	0,0,0,0,0	choice5	$M_5'^3$		
SYS	$M_5'^3$	0	choice6	req	req(C18)	Do you want a hotel in Amsterdam?
USR					con(C18=yes)	**Yes**
ASR						*Yes*
SYS	$M_5'^3$	4				[terminal state]
SYS	$M_2'^2$	0,0,2,0,0	choice5	$M_6'^3$		

Table C.5: *Real conversation in the CSTR travel planning spoken dialogue system using semi-learnt hierarchical dialogue control (PART 5).*

Agent	Subtask	Env. State	Choice State	Action	Dialogue Act	Utterance
SYS	$M_6^{\prime 3}$	19,0,0,0	choice8	req	req(C19)	Tell me your hotel information.
USR					pro(C20=cheap,C19=downtown)	**I would like a cheap hotel in downtown**
ASR						*Yes*
SYS	$M_6^{\prime 3}$	19,2,3,0	choice10	mec	mec(C20=cheap,C19=downtown)	Did you say a cheap hotel in downtown?
USR					con(C20=yes,C19=yes)	**Yes**
ASR						*Yes*
SYS	$M_6^{\prime 3}$	21,4,4,0	choice8	req	req(C21)	What is your preferred hotel chain?
USR					pro(C21=any hotel)	**Any hotel**
ASR						*Any_hotel*
SYS	$M_6^{\prime 3}$	21,4,4,2	choice9	sec	sec(C21=any hotel)	Did you say any hotel brand?
USR					con(C21=yes)	**Yes**
ASR						*Yes*
SYS	$M_6^{\prime 3}$	19,4,4,4				[terminal state]
SYS	$M_2^{\prime 2}$	0,0,2,2,0	choice5	dbq+sta		Please wait while I retrieve information.
SYS	$M_2^{\prime 2}$	2,0,2,2,0	choice5	$M_7^{\prime 3}$		
SYS	$M_7^{\prime 3}$	0,0,0,0	choice14	pre+ofr	pre(db tuples)+ofr(C22)	I found 2 hotels in Amsterdam downtown. Choice 1: Die Port Van Cleve, it is a 4 stars hotel 0.1 kilometers from downtown. the price per adult per night is 90 pounds and 78 pence. Choice 2: Swissotel Amsterdam, it is a 4 stars hotel 0.2 kilometers from downtown. the price per adult per night is 99 pounds and

Table C.6: *Real conversation in the CSTR travel planning spoken dialogue system using semi-learnt hierarchical dialogue control (PART 6).*

Agent	Subtask	Env. State	Choice State	Action	Dialogue Act	Utterance
						43 pence. Which choice do you prefer?
USR					pro(C22=one)	**Choice one**
ASR						*Choice one*
SYS	$M_7'^3$	0,0,1,2	choice15	sec	sec(C22=one)	Did you say choice one?
USR					con(C22=yes)	**Yes**
ASR						*Yes*
SYS	$M_7'^3$	0,0,1,4	choice16	ack	ack(G03)	I have added this hotel to your itinerary.
SYS	$M_7'^3$	1,0,1,4				[terminal state]
SYS	$M_2'^2$	2,0,2,2,2				[terminal state]
SYS	$M_0'^0$	4,1,2,2,0,0		$M_3'^2$		
SYS	$M_3'^2$	0,0,0,0,0,0		$M_8'^3$		
SYS	$M_8'^3$	0	choice6	req	req(C23)	Do you want a car in Amsterdam?
USR					con(C23=yes)	**Yes**
ASR						*Yes*
SYS	$M_8'^3$	4				[terminal state]
SYS	$M_3'^2$	0,0,3,0,0,0	choice5	$M_9'^3$		
SYS	$M_9'^3$	24,0,0,0,0,0	choice8	req	req(C24)	Tell me your car information.
USR					pro(C24=compact,C25=airport)	**I would like a compact car near the airport**
ASR						*I would like a compact car near the airport*
SYS	$M_9'^3$	24,2,2,0,0,0	choice10	mec	mec(C24=compact,C25=airport)	Did you say a compact car near the airport?
USR					con(C24=yes,C25=yes)	**Yes**
ASR						*Yes hertz*

Table C.7: *Real conversation in the CSTR travel planning spoken dialogue system using semi-learnt hierarchical dialogue control (PART 7).*

Agent	Subtask	Env. State	Choice State	Action	Dialogue Act	Utterance
SYS	M'^{3}_{9}	24,2,2,0,0,0	choice10	mec	mec(C24=compact,C25=airport)	Did you say a compact car near the airport?
USR					con(C24=yes,C25=yes)	**Yes**
ASR						*Yes*
SYS	M'^{3}_{9}	26,4,4,0,0,0	choice8	req	req(C26)	When do you want to pick it up?
USR					pro(C26=january 22,C27=07 pm)	**On january twenty second at seven p.m.**
ASR						*I january twenty second at seven p.m.*
SYS	M'^{3}_{9}	26,4,4,2,3,0	choice10	mec	mec(C26=january 22,C27=07 pm)	Did you say a car on january twenty second at seven pm?
USR					con(C26=yes,C27=yes)	**Yes**
ASR						*Yes*
SYS	M'^{3}_{9}	28,4,4,4,4,0	choice8	req	req(C28)	How many days do you want to rent it?
USR					pro(C28=three)	**Three days**
ASR						*Yes please*
SYS	M'^{3}_{9}	28,4,4,4,4,0	choice8	req	req(C28) rep(C28=three)	How many days do you want to rent it? For example you can say: a car for two days.
USR						**Three days**
ASR						*Three days any_rental_company*
SYS	M'^{3}_{9}	28,4,4,4,4,2	choice9	sec	sec(C28=three)	Did you say a car for three days?
USR					con(C28=yes)	**Yes**
ASR						*Yes*

Table C.8: *Real conversation in the CSTR travel planning spoken dialogue system using semi-learnt hierarchical dialogue control (PART 8).*

Agent	Subtask	Env. State	Choice State	Action	Dialogue Act	Utterance
SYS	M'^{3}_{9}	24,4,4,4,4,4				[terminal state]
SYS	M'^{2}_{3}	0,0,2,2,0,0	choice5	M'^{3}_{A}		
SYS	M'^{3}_{A}	2	choice12	sec	sec(C29=any_company)	Did you say a car rented from any rental company?
USR					con(C29=yes)	**Yes**
ASR						*Yes*
SYS	M'^{3}_{A}	4				[terminal state]
SYS	M'^{2}_{3}	0,0,2,2,2,0	choice5	dbq+sta		Please wait while I retrieve information.
SYS	M'^{2}_{3}	2,0,2,2,2,0	choice5	M'^{3}_{B}		
SYS	M'^{3}_{B}	0,0,0,0	choice14	pre+ofr	pre(db tuples)+ofr(C30)	I found 2 compact cars in Amsterdam airport rented from auto Europe. Choice 1: the price per day is 60 pounds and 8 pence. Choice 2: the price per day is 81 pounds and 13 pence. Which choice do you prefer?
USR					pro(C30=two)	**Choice two**
ASR						*Choice two*
SYS	M'^{3}_{B}	0,0,1,2	choice15	sec	sec(C30=two)	Did you say choice two?
USR					con(C30=yes)	**Yes**
ASR						*Yes*
SYS	M'^{3}_{B}	0,0,1,4	choice16	ack	ack(G04)	I have added this car to your itinerary.
SYS	M'^{3}_{B}	1,0,1,4				[terminal state]
SYS	M'^{2}_{3}	2,0,2,2,2,2				[terminal state]

Table C.9: *Real conversation in the CSTR travel planning spoken dialogue system using semi-learnt hierarchical dialogue control (PART 9).*

Agent	Subtask	Env. State	Choice State	Action	Dialogue Act	Utterance
SYS	$M_0'^0$	5,1,2,2,2,0	choice2	$M_4'^2$		
SYS	$M_4'^2$	0,0,0,0	choice5	$M_C'^3$		
SYS	$M_C'^3$	0	choice8	req	req(C31)	Do you want a summary of your trip?
USR					con(C31=yes)	**Yes**
ASR						*Yes*
SYS	$M_C'^3$	4				[terminal state]
SYS	$M_4'^2$	0,0,2,0	choice5	dbq+sta		Let me retrieve your information.
SYS	$M_4'^2$	2,0,2,0	choice5	$M_D'^3$		
SYS	$M_D'^3$	0,0,0,0	choice14	pre+ofr	pre(db tuples)+ofr(C32)	All right. You have a flight from Edinburgh to Amsterdam with British Airways on january twenty second departing at six fourty five a.m. and arriving at eleven twenty a.m. It returns on february first departing at four forty five p.m. and arriving at seven fifty p.m. It costs 147 pounds and 60 pence. In Amsterdam you have a single bedroom in the Die Port Van Cleve downtown hotel. It costs 90 pounds and 78 pence per night. Also you have a compact car in Amsterdam airport with auto Europe for three days. It costs 81 pounds and 13 pence per day. The total cost of your travel is 1 thousand 208 pounds and 1 pence.

Table C.10: *Real conversation in the CSTR travel planning dialogue system using semi-learnt hierarchical dialogue control (PART 10).*

Agent	Subtask	Env. State	Choice State	Action	Dialogue Act	Utterance
						Do you want to book this trip?
USR					con(C32=yes)	**Yes**
ASR						*Yes*
SYS	M'^3_D	0,0,1,4	choice16	ack	ack(G05)	All right, your trip has been booked.
SYS	M'^3_D	1,0,1,4				[terminal state]
SYS	M'^2_4	2,0,2,2				[terminal state]
SYS	M'^0_0	5,1,2,2,2,2	choice3	clo	clo()	Thanks for calling the CSTR travel planning system. Good bye!
SYS	M'^0_0	5,2,2,2,2,2				[terminal state]

Abbreviations:

SYS=CSTR travel planning spoken dialogue system

CSTR=Centre for speech technology research

USR=Participant in the evaluation of the dialogue system

ASR=Automatic speech recognition hypothesis

M'^i_j=Induced dialogue subtask

Env. State=Environment state in SMDP M^i_j

Choice State=Choice state in HAM for the current subtask

SMDP=Semi-Markov decision process

HAM=Hierarchical abstract machine

References

Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., and Stent, A. (2001a). Towards conversational human-computer interaction. *AI Magazine*, 22(4):27–37.

Allen, J., Ferguson, G., and Stent, A. (2001b). An architecture for more realistic conversational systems. In *Intelligent User Interfaces Conference (IUI)*, pages 1–8, Santa Fe, NM, USA.

Andre, D. (2003). *Programmable Reinforcement Learning Agents*. PhD thesis, University of California at Berkeley.

Andre, D. and Russell, S. (2000). Programmable reinforcement learning agents. In *Neural Information Processing Systems Conference (NIPS)*, pages 1019–1025, Cambridge, MA.

Andre, D. and Russell, S. (2002). State abstraction for programmable reinforcement learning agents. In *AAAI Conference on Artificial Intelligence*, pages 119–125, Alberta, Canada.

Asadi, M. and Huber, M. (2007). Effective control knowledge transfer through learning skill and representation hierarchies. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2054–2059, Hyderabad, India.

Atrash, A. and Pineau, J. (2006). Efficient planning and tracking in POMDPs with large observation spaces. In *AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*, pages 7–12, Boston, MA, USA.

Austin, J. (1962). *How to Do Things with Words*. Cambridge University Press.

Barto, A. and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13(1-2):41–77.

Benson, S. and Nilsson, N. (1996). *Machine Intelligence 14: Applied Machine Intelligence*, chapter Reacting, Planning and Learning in an Autonomous Agent, pages 29–62. Oxford University Press.

Bertsekas, D. and Tsitsiklis, J. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Bohus, D. (2007). *Error Awareness and Recovery in Task-Oriented Spoken Dialog Systems*. PhD thesis, Carnegie Mellong University.

Bohus, D. and Rudnicky, A. (2003). Ravenclaw: Dialogue management using hierarchical task decomposition and an expectation agenda. In *European Conference on Speech Communication and Technology (Eurospeech)*, pages 597–600, Geneva, Switzerland.

Bohus, D. and Rudnicky, A. (2005a). Constructing accurate beliefs in spoken dialogue systems. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 272–277, San Juan, Puerto Rico.

Bohus, D. and Rudnicky, A. (2005b). Sorry, i didn't catch that! - an investigation of non-understanding errors and recovery strategies. In *Workshop on Discourse and Dialogue (SIGDIAL)*, pages 128–143, Lisbon, Portugal.

Bohus, D. and Rudnicky, A. (2006). A 'K hypothesis + other' belief updating model. In *AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*, pages 13–18, Boston, MA, USA.

Boutilier, C., Dean, T., and Hanks, S. (1999). Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11(1):1–94.

Bradtke, S. and Duff, M. (1994). Reinforcement learning methods for continuous-time Markov decision processes. In *Neural Information Processing Systems Conference (NIPS)*, pages 393–400, Denver, CO, USA.

Cheyer, A. and Martin, D. (2001). The open agent architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1-2):143–148.

Chu-Carroll, J. (1999). Form-based reasoning for mixed-initiative dialogue management in information-query systems. In *European Conference on Speech Communication and Technology (Eurospeech)*, pages 1519–1522, Budapest, Hungry.

Chu-Carroll, J. and Nickerson, J. (2000). Evaluating automatic dialogue strategy adaptation for a spoken dialogue system. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 202–209, Seattle, WA.

Chung, G. (2004). Developing a flexible spoken dialog system using simulation. In *International Conference on Computational Linguistics (ACL)*, pages 63–70, Barcelona, Spain.

Clark, H. (1996). *Using Language*. Cambridge University Press.

Cuayáhuitl, H., Renals, S., Lemon, O., and Shimodaira, H. (2005). Human-computer dialogue simulation using hidden Markov models. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 290–295, San Juan, Puerto Rico.

Cuayáhuitl, H., Renals, S., Lemon, O., and Shimodaira, H. (2006a). Learning dialogue strategies using reinforcement learning with reduced state-action spaces. In *INTERSPEECH*, pages 469–472, Pittsburgh, PA, USA.

Cuayáhuitl, H., Renals, S., Lemon, O., and Shimodaira, H. (2006b). Reinforcement learning of dialogue strategies using hierarchical abstract machines. In *IEEE Workshop on Spoken Language Technology (SLT)*, pages 182–185, Palm Beach, Aruba.

Cuayáhuitl, H., Renals, S., Lemon, O., and Shimodaira, H. (2007). Hierarchical dialogue optimization using semi-Markov decision processes. In *INTERSPEECH*, pages 2693–2696, Antwerp, Belgium.

Currie, K. and Tate, A. (1991). O-plan: the Open planning architecture. *Artificial Intelligence*, 52(1):49–86.

Dayan, P. and Hinton, G. (1992). Feudal reinforcement learning. In *Neural Information Processing Systems Conference (NIPS)*, pages 271–278, San Francisco, CA, USA.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

Denecke, M., Dohsaka, K., and Nakano, M. (2004). Fast reinforcement learning of dialogue policies using stable function approximation. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1–11, Jeju, Korea.

Dietterich, T. (1999). State abstraction in MAXQ hierarchical reinforcement learning. In *Neural Information Processing Systems Conference (NIPS)*, pages 994–1000, Denver, CO, USA.

Dietterich, T. (2000a). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13(1):227–303.

Dietterich, T. (2000b). An overview of MAXQ hierarchical reinforcement learning. In *Symposium on Abstraction, Reformulation, and Approximation (SARA)*, pages 26–44, HorseshoeBay, TX, USA.

Dzeroski, S., Raedt, L., and Driessens, K. (2001). Relational reinforcement learning. *Machine Learning*, 43:7–52.

Eckert, W., Levin, E., and Pieraccini, R. (1997). User modeling for spoken dialogue system evaluation. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 80–87, Santa Barbara, CA, USA.

English, M. and Heeman, P. (2005). Learning mixed initiative dialogue strategies by using reinforcement learning on both conversants. In *Human Language Technology Conference (HLT)*, pages 1011–1018, Vancouver, Canada.

Filisko, E. and Seneff, S. (2005). Developing city name acquisition strategies in spoken dialogue systems via user simulation. In *Workshop on Discourse and Dialogue (SIGDIAL)*, pages 144–155, Lisbon, Portugal.

Filisko, E. and Seneff, S. (2006). Learning decision models in spoken dialogue systems via user simulation. In *AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*, pages 19–24, Boston, MA, USA.

Foka, A. and Trahanias, P. (2007). Real-time hierarchical POMDPs for autonomous robot navigation. *Robotics and Autonomous Agents*, 55(7):561–571.

Frampton, M. and Lemon, O. (2005). Reinforcement learning of dialogue strategies using the user's last dialogue act. In *Workshop on Knowledge and Reasoning in Practical Dialogue Systems (IJCAI)*, pages 83–90, Edinburgh, Scotland.

Frampton, M. and Lemon, O. (2006). Learning more effective dialogue strategies using limited dialogue move features. In *International Conference on Computational Linguistics (ACL)*, pages 185–192, Sydney, Australia.

Frampton, M. and Lemon, O. (2008). Using dialogue acts to learn better repair strategies for spoken dialogue systems. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5045–5048, Las Vegas, NV, USA.

Gasic, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., Yu, K., and Young, S. (2008). Training and evaluation of the HIS POMDP dialogue system in noise. In *Workshop on Discourse and Dialogue (SIGDIAL)*, pages 112–119, Columbus, Ohio, USA.

Georgila, K., Henderson, J., and Lemon, O. (2005a). Learning user simulations for information state update dialogue systems. In *INTERSPEECH*, pages 893–896, Lisbon, Portugal.

Georgila, K., Henderson, J., and Lemon, O. (2006). User simulation for spoken dialogue systems: Learning and evaluation. In *INTERSPEECH*, pages 267–659, Pittsburgh, PA, USA.

Georgila, K., Lemon, O., and Henderson, J. (2005b). Automatic annotation of communicator dialogue data for learning dialogue strategies and user simulations. In *Workshop on the Semantics and Pragmatics of Dialogue (DIALOR)*, Nancy, France.

Ghavamzadeh, M. and Mahadevan, S. (2001). Continuous-time hierarchical reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 186–193, Williams College, MA, USA.

Ghavamzadeh, M. and Mahadevan, S. (2007). Hierarchical average reward reinforcement learning. *Journal of Machine Learning Research*, 8:2629–2669.

Goddeau, D., Meng, H., Polifroni, J., Seneff, S., and Busayapongchai, S. (1996). A form-based dialogue manager for spoken language applications. In *International Conference on Speech and Language Processing (ICSLP)*, pages 701–704, Philadelphia, PA, USA.

Goddeau, D. and Pineau, J. (2000). Fast reinforcement learning of dialogue strategies. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages II1233–II1236, Istambul, Turkey.

Gordon, G. (2000). Reinforcement learning with function approximation converges to a region. In *Neural Information Processing Systems Conference (NIPS)*, pages 1040–1046, Denver, CO, USA.

Grosz, B. and Sidner, C. (1986). Attention, intentions and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Gruber, T. (1993). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5-6):907–928.

Hansen, E. and Zhou, R. (2003). Synthesis of hierarchical finite-state controllers for POMDPs. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 113–122, Trento, Italy.

Heeman, P. (2007). Combining reinforcement learning with information-state update rules. In *Human Language Technology Conference (HLT)*, pages 268–275, Rochester, NY, USA.

Henderson, J. and Lemon, O. (2008). Mixture model POMDPs for efficient handling of uncertainty in dialogue management. In *International Conference on Computational Linguistics (ACL)*, pages 73–76, Columbus, Ohio, USA.

Henderson, J., Lemon, O., and Georgila, K. (2005). Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In *Workshop on Knowledge and Reasoning in Practical Dialogue Systems (IJCAI)*, pages 68–75, Edinburgh, Scotland.

Henderson, J., Lemon, O., and Georgila, K. (2008). Hybrid reinforcement/supervised learning of dialogue policies from from fixed data sets. volume 34, pages 487–511.

Hengst, B. (2003). *Discovering Hierarchy in Reinforcement Learning*. PhD thesis, University of New South Wales.

Horvitz, E. and Paek, T. (1999). A computational architecture for conversation. In *International Conference on User Modelling (UM)*, pages 201–210, Banff, Canada.

Horvitz, E. and Paek, T. (2000). Deeplistener: Harnessing expected utility to guide clarification dialog in spoken language systems. In *International Conference on Speech and Language Processing (ICSLP)*, pages 226–229, Beiging, China.

Howard, R. (1971). *Dynamic Probabilistic Systems: Semi-Markov and Decision Processes (Volume II)*. Dover Publications, Inc.

Huang, X., Acero, A., and Hon, H. (2001). *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall.

Hurtado, L., Griol, D., Sanchis, E., and Segarra, E. (2007). A statistical user simulation technique for the improvement of a spoken dialog system. In *Iberoamerican Congress on Pattern Recognition (CIARP)*, pages 743–752, Viña del Mar, Chile.

Jaakkola, T., Jordan, M., and Singh, S. (1994). Oh the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201.

Jong, N. and Stone, P. (2005). State abstraction discovery from irrelevant state variables. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 752–757, Edinburgh, Scotland.

Jonsson, A. (2008). *Hierarchical Decomposition in Reinforcement Learning*. VDM Verlag Dr. Muller.

Jonsson, A. and Barto, A. (2000). Automated state abstraction for options using the U-Tree algorithm. In *Neural Information Processing Systems Conference (NIPS)*, pages 1054–1060, Denver, CO, USA.

Jurafsky, D. and Martin, J. (2008). *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall.

Kaelbling, . (1993). Hierarchical reinforcement learning: Preliminary results. In *International Conference on Machine Learning (ICML)*, pages 167–163, San Francisco, CA, USA.

Kaelbling, L., Littman, M., and Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

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

Karlsson, R. (1997). *Learning to Solve Multiple Goals*. PhD thesis, University of Rochester.

Konidaris, G. and Barto, A. (2007). Building portable options: Skill transfer in reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 895–900, Hyderabad, India.

Larsson, S. and Traum, D. (2000). Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 5(3-4):323–340.

Lemon, O., Georgila, K., and Henderson, J. (2006a). Evaluating efectiveness and portability of reinforcement learned dialogue strategies with real users: The TALK TownInfo evaluation. In *IEEE Workshop on Spoken Language Technology (SLT)*, pages 178–181, Palm Beach, Aruba.

Lemon, O., Kallirroi, G., and Stuttle, M. (2005). D4.2: Showcase exhibiting reinforcement learning for dialogue strategies in the in-car domain. Technical Report Talk Project, Deliverable 4.2, University of Edinburgh.

Lemon, O., Liu, X., Shapiro, D., and Tollander, C. (2006b). Hierarchical reinforcement learning of dialogue policies in a development environment for dialogue systems: REALL-DUDE. In *Workshop on the Semantics and Pragmatics of Dialogue (BRANDIAL)*, pages 185–186, Postdam, Germany.

Levin, E. and Pieraccini, R. (1997). A stochastic model of computer-human interaction for learning dialog strategies. In *European Conference on Speech Communication and Technology (Eurospeech)*, pages 1883–1886, Rhodes, Greece.

Levin, E., Pieraccini, R., and Eckert, W. (1998). Using Markov decision process for learning dialog strategies. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 201–204, Istanbul, Turkey.

Levin, E., Pieraccini, R., and Eckert, W. (2000). A stochastic model of human machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23.

Lin, B. and Lee, L. (2001). Computer-aided analysis and design for spoken dialogue systems based on quantitative simulations. *IEEE Transactions on Speech and Audio Processing*, 8(5):534–548.

Litman, D. and Allen, J. (1987). A plan recognition model for subdialogues in conversations. *Cognitive Science*, 11:163–200.

Litman, D., Kearns, M., Singh, S., and Walker, M. (2000). Automatic optimization of dialogue management. In *International Conference on Computational Linguistics (COLING)*, pages 502–508, Saarbrucken, Germany.

Litman, D. and Pan, S. (2002). Designing and evaluating an adaptive spoken dialogue system. *User Modeling and User-Adapted Interaction*, 12(2/3):111–137.

López-Cózar, R., Callejas, Z., and McTear, M. (2008). Testing the performance of spoken dialogue systems by means of an artificial user. *Artificial Intelligence Review*, 26(4):291–323.

López-Cózar, R., De la Torre, A., Segura, J., and Rubio, J. (2003). Assessment of dialogue systems by means of a new simulation technique. *Speech Communication*, 40(3):387–407.

Mahadevan, S., Ghavamzadeh, M., Rohanimanesh, K., and Theocarous, G. (2004). *Handbook of Learning and Approximate Dynamic Programming*, chapter Hierarchical Approaches to Concurrency, Multiagency, and Partial Observability, pages 285–310. John Wiley & Sons.

Mahadevan, S., Marchalleck, N., Das, T., and Gosavi, A. (1997). Self-improving factory simulation using continuous-time average-reward reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 202–210, Nashville, TN, USA.

Makar, R. and Mahadevan, S. (2001). Hierarchical multi-agent reinforcement learning. In *International Conference on Autonomous agents (Agents)*, pages 246–253, Montreal, Canada.

Marthi, B. (2006). *Concurrent Hierarchical Reinforcement Learning*. PhD thesis, University of California at Berkeley.

Marthi, B., Russell, S., and Andre, D. (2006). A compact, hierarchical Q-function decomposition. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, Cambridge, MA, USA.

Marthi, B., Russell, S., Latham, D., and Guestrin, C. (2005). Concurrent hierarchical reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 779–785, Edinburgh, Scotland.

McGovern, A. (2002). *Autonomous Discovery of Temporal Abstractions from Interaction with an Environment*. PhD thesis, University of Massachusetts Amherst.

McTear, M. (1998). Modelling spoken dialogues with state transition diagrams: Experiences with the CSLU toolkit. In *International Conference on Speech and Language Processing (ICSLP)*, pages 1223–1226, Sidney, Australia.

McTear, M. (2004). *Spoken Dialogue Technology: Toward the Conversational User Interface*. Springer.

Mitchell, T. (2004). *Machine Learning*. McGraw-Hill.

Moriarty, D., Schultz, A., and Grefenstette, J. (1999). Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276.

Nilsson, N. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1:139–158.

Ohta, M., Kumada, Y., and Noda. (2003). Using suitable action selection rule in reinforcement learning. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 4358–4363, Washington, DC, USA.

Paek, T. (2006). Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment. In *Workshop on Dialogue on Dialogues - Multidisciplinary Evaluation of Advanced Speech-based Interacive Systems (INTERSPEECH)*, Pittsburgh, PA, USA.

Paek, T. and Chickering, D. M. (2005). The Markov assumption in spoken dialogue management. In *Workshop on Discourse and Dialogue (SIGDIAL)*, pages 35–44, Lisbon, Portugal.

Paek, T. and Horvitz, E. (2000). Conversation and action under uncertainty. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 455–464, San Francisco, CA, USA.

Paek, T. and Pieraccini, R. (2008). Automating spoken dialogue management design using machine learning: An industry perspective. volume 50, pages 716–729.

Parr, R. (1998). *Hierarchical Control and Learning for Markov Decision Processes*. PhD thesis, University of California at Berkeley.

Parr, R. and Russell, S. (1997). Reinforcement learning with hierarchies of machines. In *Neural Information Processing Systems Conference (NIPS)*, pages 1043–1049, Denver, CO, USA.

Pieraccini, R., Caskey, S., Dayanidhi, K., Carpenter, B., and Phillips, M. (2001). ETUDE, a recursive dialogue manager with embedded user interface patterns. In

IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pages 244–247, Madonna di Campiglio, Italy.

Pietquin, O. (2004). *A Framework for Unsupervised Learning of Dialogue Strategies*. PhD thesis, Faculté Polytechnique de Mons.

Pietquin, O. (2006). Consistent goal-directed user model for realistic man-machine task-oriented spoken dialogue simulation. In *IEEE International Conference on Multimedia and Expo*, pages 425–428, Toronto, Canada.

Pietquin, O. (2007). Learning to ground in spoken dialogue systems. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 165–168, Hawaii, USA.

Pietquin, O. and Dutoit, T. (2006). A probabilistic framework for dialogue simulation and optimal strategy learning. *IEEE Transactions on Speech and Audio Processing*, 14(2):589–599.

Pietquin, O. and Renals, S. (2002). ASR system modeling for automatic evaluation and optimization of dialogue systems. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 46–49, Orlando, FL, USA.

Pineau, J. (2004). *Tractable Planning Under Uncertainty: Exploiting Structure*. PhD thesis, Carnegie Mellon University.

Pineau, J., Gordon, G., and Thrun, S. (2006). Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380.

Pineau, J., Roy, N., and Thrun, S. (2001). A hierarchical approach to POMDP planning and execution. In *Workshop on Hierarchy and Memory in Reinforcement Learning (ICML)*, William College, Massachusetts.

Precup, D. (2000). *Temporal Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts Amherst.

Prommer, T., Holzapfel, H., and Waibel, A. (2006). Rapid simulation-driven reinforcement learning of multimodal dialog strategies in human-robot interaction. In *INTERSPEECH*, pages 1918–1921, Pittsburgh, PA, USA.

Putterman, M. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Reichman, R. (1978). Conversational coherency. *Cognitive Science*, 2(4):283–327.

Rich, C. and Sidner, C. (1998). Collagen: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3/4):315–350.

Rieser, V. and Lemon, O. (2006a). Cluster-based user simulations for learning dialogue strategies. In *INTERSPEECH*, pages 1766–1769, Pittsburgh, PA, USA.

Rieser, V. and Lemon, O. (2006b). Using logistic regression to initialize reinforcement-learning-based dialogue systems. In *IEEE Workshop on Spoken Language Technology (SLT)*, pages 190–193, Palm Beach, Aruba.

Rieser, V. and Lemon, O. (2007). Learning dialogue strategies for interactive database search. In *INTERSPEECH*, pages 2689–2692, Antwerp, Belgium.

Rieser, V. and Lemon, O. (2008). Learning effective multimodal dialogue strategies from Wizard-of-Oz data: Bootstrapping and evaluation. In *International Conference on Computational Linguistics (ACL)*, pages 638–646, Columbus, Ohio, USA.

Roy, N., Pineau, J., and Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning. In *International Conference on Computational Linguistics (ACL)*, pages 93–100, Hong Kong.

Rudnicky, A., Thayer, E., Constantinides, P., Tchou, C., Shern, R., Lenzo, K., W., X., and Oh, A. (1999). Creating natural dialogs in the Carnegie Mellon Communicator system. In *European Conference on Speech Communication and Technology (Eurospeech)*, pages 1531–1534, Budapest, Hungary.

Rudnicky, A. and Wu, W. (1999). An agenda-based dialogue management architecture for spoken language systems. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 337–340, Keystone, Colorado, USA.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education.

Ryan, M. (2002). *Hierarchical Reinforcement Learning: A Hybrid Approach*. PhD thesis, University of New South Wales.

Sacerdoti, E. (1975). The nonlinear nature of plans. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 206–215, Tbilisi, Georgia, URRS.

Schatzmann, J., Georgila, K., and Young, S. (2005a). Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Workshop on Discourse and Dialogue (SIGDIAL)*, Lisbon, Portugal.

Schatzmann, J., Stuttle, M. N., Weilhammer, K., and Young, S. (2005b). Effects of the user model on simulation-based learning of dialogue strategies. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 220–225, San Juan, Puerto Rico.

Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., and Young, S. (2007a). Agenda-based user simulations for bootstrapping a POMDP dialogue system. In *Human Language Technology Conference (HLT)*, pages 149–152, Rochester, NY.

Schatzmann, J., Thomson, B., and Young, S. (2007b). Error simulation for training statistical dialogue systems. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 526–531, Kyoto, Japan.

Schatzmann, J., Thomson, B., and Young, S. (2007c). Statistical user simulation with a hidden agenda. In *Workshop on Discourse and Dialogue (SIGDIAL)*, pages 273–282, Antwerp, Belgium.

Schatzmann, J., Weilhammer, K., Stuttle, M., and Young, S. (2006). A survey on statistical user simulation techniques for reinforcement learning of dialogue management strategies. *Knowledge Engineering Review*, 21(2):97–126.

Scheffler, K. (2002). *Automatic Design of Spoken Dialogue Systems*. PhD thesis, Cambridge University.

Scheffler, K. and Young, S. (2000). Probabilistic simulation of human-machine dialogues. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1217–1220, Istanbul, Turkey.

Scheffler, K. and Young, S. (2001). Corpus-based simulation for automatic strategy learning and evaluation. In *Workshop on Adaptation in Dialogue Systems (NAACL)*, Pittsburgh, PA, USA.

Scheffler, K. and Young, S. (2002). Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Human Language Technology Conference (HLT)*, pages 12–19, San Diego, CA, USA.

Searle, J. (1969). *Speech Acts*. Cambridge University Press.

Seneff, S. and Polifroni, J. (2000). Dialogue management in the Mercury flight reservation system. In *ANLP/NAACL, Workshop on Conversational Systems*, pages 11–16, Seattle, WA, USA.

Shapiro, D. and Langley, P. (2002). Separating skills from preference: Using learning to program by reward. In *International Conference on Machine Learning (ICML)*, pages 570–577, Sydney, Australia.

Singh, S. (1992). Reinforcement learning with a hierarchy of abstract models. In *AAAI Conference on Artificial Intelligence*, pages 202–207, San Jose, CA, USA.

Singh, S., Kearns, M., Litman, D., and Walker, M. (1999). Reinforcement learning for spoken dialogue systems. In *Neural Information Processing Systems Conference (NIPS)*, pages 956–962, Denver, CO, USA.

Singh, S., Litman, D., Kearns, M., and Walker, M. (2002). Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16:105–133.

Singh, S. and Sutton, R. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158.

Skantze, G. (2007). *Error Handling in Spoken Dialogue Systems: Managing Uncertainty, Grounding and Miscommunication*. PhD thesis, KTH - Royal Institute of Technology.

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

Staab, S. and Studer, R. (2004). *Handbook on Ontologies*. Springer.

Stolle, M. and Precup, D. (2002). Learning options in reinforcement learning. In *Symposium on Abstraction, Reformulation, and Approximation (SARA)*, pages 212–223, Kananaskis, Alberta, Canada.

Stone, P., Sutton, R., and Kuhlmann, G. (2005). Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behaviour*, 13(3):165–188.

Sutton, R. and Barto, A. (1998). *Reinforcement Learing: An Introduction*. MIT Press.

Sutton, R., Precup, D., and Singh, S. (1999). Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211.

Tadepalli, P., Givan, R., and Driessens, K. (2004). Relational reinforcement learning: An overview. In *Workshop on Relational Reinforcement Learning - ICML*, pages 1–9, Banff, Canada.

Taylor, M. and Stone, P. (2007). Cross-domain transfer for reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 879–886, Corvallis, OR, USA.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68.

Theocarous, G. (2002). *Hierarchical Learning and Planning in Partially Observable Markov Decision Processes*. PhD thesis, Michigan State University.

Theocarous, G., Murphy, K., and Kaelbking, L. (2004). Representing hierarchical POMDPs as DBNs for multi-scale robot localization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1045–1051, New Orleans, LA, USA.

Thomson, B., Schatzmann, J., and Young, S. (2008). Bayesian update of dialogue state for robust dialogue systems. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4937–4940, Las Vegas, USA.

Toney, D. (2007). *Evolutionary Reinforcement Learning of Spoken Dialogue Strategies*. PhD thesis, University of Edinburgh.

Toney, D., Moore, J., and Lemon, O. (2006a). Developing conversational interfaces with XCS. In *Workshop on Learning Classifier Systems (GECCO)*, Seattle, WA, USA.

Toney, D., Moore, J., and Lemon, O. (2006b). Evolving optimal inspectable strategies for spoken dialogue systems. In *Human Language Technology Conference (HLT)*, pages 173–176, New York, NY, USA.

Torres, F., Sanchis, E., and Segarra, E. (2008). User simulation in a stochastic dialog system. *Computer Speech and Language*, 22(3):230–255.

Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155.

Uther, W. (2002). *Tree Based Hierarchical Reinforcement Learning*. PhD thesis, Carnegie Mellon University.

Walker, M., Passonneau, R., and Boland, J. (2001). Quantitative and qualitative evaluation of DARPA Communicator spoken dialogue systems. In *International Conference on Computational Linguistics (ACL)*, pages 515–522.

Walker, M. (2000). An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, 12:387–416.

Walker, M., Kamm, C., and Litman, D. (2000). Towards developing general models of usability with PARADISE. *Natural Language Engineering*, 6(3):363–377.

Walker, M., Litman, D., Kamm, C., and Abella, A. (1997). PARADISE: A framework for evaluating spoken dialogue agents. In *International Conference on Computational Linguistics (ACL)*, pages 271–280, Madrid, Spain.

Walker, M. and Passonneau, R. (2001). DATE: A dialogue act tagging scheme for evaluation of spoken dialogue systems. In *Human Language Technology Conference (HLT)*, pages 1–8, San Diego, CA, USA.

Walker, M., Rudnicky, A., Prasad, R., Aberdeen, J., Bratt, E., Garofolo, J., Hastie, H., Le, A., Pellom, B., Potamianos, A., Passonneau, R., Roukos, S., Sanders, G., Seneff, S., and Stallard, D. (2002). DARPA communicator: Cross-system results for the 2001 evaluation. In *International Conference on Speech and Language Processing (ICSLP)*, pages 273–276, Colorado, CO, USA.

Ward, W. (1994). Extracting information from spontaneous speech. In *International Conference on Speech and Language Processing (ICSLP)*, pages 18–22, Yokohama, Japan.

Watkins, C. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College.

Watkins, C. and Dayan, P. (1992). Technical note Q-learning. *Machine Learning*, 8(3-4):279–292.

Whiteson, S. and Stone, P. (2006). Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, 7:877–917.

Williams, J. (2006). *Partially Observable Markov Decision Processes for Spoken Dialogue Management*. PhD thesis, Cambridge University.

Williams, J. (2007a). A method for evaluating and comparing user simulations: The Cramer-Von Misses divergence. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 508–513, Kyoto, Japan.

Williams, J. (2007b). Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422.

Williams, J. (2007c). Scaling POMDPs for spoken dialog management. *IEEE Transactions on Speech and Audio Processing*, 15(7):2116–2129.

Williams, J. (2007d). Using particle filters to track dialogue state. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 502–507, Kyoto, Japan.

Williams, J. (2008a). The best of both worlds: Unifying conventional dialog systems and POMDPs. In *INTERSPEECH*, Brisbane, Australia.

Williams, J. (2008b). Integrating expert knowledge into POMDP optimization for spoken dialogue systems. In *AAAI Workshop on Advancements in POMDP Solvers*, Chicago, USA.

Wilson, A., Fern, A., Ray, S., and Tadepalli, P. (2007). Multi-task reinforcement learning: A hierarchical Bayesian approach. In *International Conference on Machine Learning (ICML)*, pages 1015–1022, Corvallis, OR, USA.

Yamagishi, J., Zen, H., Toda, T., and Tokuda, K. (2007). Speaker-independent HMM-based speech synthesis system - HTS-2007 system for the blizzard challenge 2007. In *The Blizzard Challenge*, pages 1–6, Bonn, Germany.

Young, S. (2000). Probabilistic methods in spoken dialogue systems. *Philosophical Transactions of the Royal Society (Series A)*, 358(1769):1389–1402.

Young, S. (2002). Talking to machines (statistically speaking). In *International Conference on Speech and Language Processing (ICSLP)*, pages 9–16, Denver, CO, USA.

Young, S. (2006). *The HTK Book*. Cambridge University Engineering Department.

Young, S. (2007). *ATK: An Application Toolkit for HTK*. Cambridge University Engineering Department.

Young, S., Schatzmann, J., Weilhammer, K., and Ye, H. (2007). The hidden information state approach to dialogue management. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 149–152, Honolulu, Hawaii.

Zhang, B., Cai, Q., Mao, J., and Guo, B. (2001). Planning and acting under uncertainty: A new model for spoken dialogue system. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 572–579, Seattle, WA, USA.

Zue, V. (2007). On organic interfaces. In *INTERSPEECH*, pages 1–8, Antwerp, Belgium.

Zue, V. and Glass, J. (2000). Conversational interfaces: Advances and challenges. *IEEE Transactions on Speech and Audio Processing*, 88(8):1166–1180.