Structured Probabilistic Modelling for Dialogue Management

Doctoral Dissertation by

Pierre Lison

Department of Informatics
Faculty of Mathematics and Natural Sciences
University of Oslo

Submitted for the degree of Philosophiae Doctor

August 20, 2013

Abstract

TODO

Contents

Mathematical notations

Probability distributions:

X	Random variable
$Val(X)$	Range of values for the random variable X
$P(X)$	Probability distribution for the random variable X
$P(X_1, ...X_n)$	Joint probability distribution for X_1, ... X_n
$P(X_1, ...X_n \mid Y_1, ...Y_m)$	Conditional probability distribution for X_1, ... X_n given Y_1, ... Y_m
$E(X)$	Expectation of the random variable X
$p(X)$	Probability density function (PDF) of continuous variable X
$P(\mathbf{Q} \mid \mathbf{E}=\mathbf{e})$	Posterior distribution of variables $\mathbf{Q}$ given evidence $\mathbf{E}=\mathbf{e}$
$U(\mathbf{Q} \mid \mathbf{E}=\mathbf{e})$	Utility distribution of variables $\mathbf{Q}$ given evidence $\mathbf{E}=\mathbf{e}$
$\boldsymbol{\theta}_X$	Parameters associated with the random variable X
$P(X\,;\boldsymbol{\theta}_X)$	Probability of X given the parameters $\boldsymbol{\theta}_X$

Graphical models:

$\mathcal{B}$	Bayesian network
$P_{\mathcal{B}}(X)$	Probability distribution for X in the Bayesian network $\mathcal{B}$
$(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$	Conditional independence of variables $\mathbf{X}$ and $\mathbf{Y}$ given $\mathbf{Z}$
$Y \rightarrow X$	Directed edge from variable Y to variable X
$parents(X)$	Parents of variable X such that $Y \rightarrow X$ for all $Y \in parents(X)$

Reinforcement learning:

s	Current state
$\mathcal{S}$	Set of possible states
s_t	State at time t
a	System action
$\mathcal{A}$	Set of possible actions
$R(s,a)$	Immediate reward of action a in state s
γ	Discount factor
h	Planning horizon
$V(s)$	Value function for state s (= expected return)
$Q(s,a)$	Action–value function for action a in state s
$\pi(s)$	MDP dialogue policy, defined as a function $\pi : \mathcal{S} \rightarrow \mathcal{A}$
o	Observation
$\mathcal{O}$	Set of possible observations

b	Belief state $b(s) = P(s)$
$\mathcal{B}$	Belief state space $\subset \Re^{\|S\|-1}$
$V(b)$	Value function for belief state b
$Q(b, a)$	Action–value function for action a in belief state b
$\pi(b)$	POMDP dialogue policy, defined as a function $\pi : \mathcal{B} \rightarrow \mathcal{A}$

Dialogue-specific variables:

u_u	User utterance
$\tilde{u}_u$	Speech recognition hypotheses for user utterance
a_u	User dialogue act
$\tilde{a}_u$	Speech understanding hypotheses for the user dialogue act
i_u	User intention
c	Interaction context
a_m	System dialogue act
u_m	System utterance

Probabilistic rules:

r	Probability or utility rule
c_i	i-th condition of a rule, expressed as a logical formula
$e_{(i,j)}$	j-th effect for condition c_i, expressed as a value assignment
$p_{(i,j)}$	Probability of effect $e_{(i,j)}$
$I_1, ...I_k$	Set of input variables of a rule
$O_1, ...O_l$	Set of output variables of a probability rule
$A_1, ...A_l$	Set of decision variables of an utility rule
$\mathbf{e}$	Conjunction of effects $e_1 \wedge ...e_n$
$\mathbf{e}(X)$	(Possibly empty) set of values specified for variable X in $\mathbf{e}$
$\mathbf{y}$	Universally quantified variables $y_1, ...y_p$ of a rule
$\mathbf{g}$	Possible grounding for the universally quantified variables $\mathbf{y}$

Miscellaneous:

$\mathbf{1}(b)$	Indicator function, with $\mathbf{1}(b) = 1$ if b is true and 0 otherwise
$\phi[a/b]$	First-order formula ϕ where all occurrences of a are replaced by b
$\|\mathcal{S}\|$	Cardinality of the set $\mathcal{S}$ (i.e. number of elements)

Chapter 1

Introduction

Spoken language is one of the most powerful system of communication at our disposal. A large part of our waking hours is spent in social interactions mediated through natural language. The pivotal role of spoken language in our daily lives is largely due to its remarkable proficiency at conveying elaborate thoughts in a robust and efficient manner.

Is it possible to exploit this basic fact to develop more user-friendly technologies? Most of our everyday activities are now relying on "smart" electronic devices of various kinds, from mobile phones to personal computers and in-car navigation systems. As these technologies gain in autonomy and sophistication, user interaction design becomes increasingly important. User interfaces should offer rich communication capabilities that can unlock the full potential of their applications, yet remain easy to understand and control. One natural way to achieve this goal is to endow computers with a capacity to understand, even in a limited manner, the communication medium that is most intuitive to human beings, namely spoken language.

The ongoing research on *spoken dialogue systems* (SDS) is precisely trying to implement this objective. A spoken dialogue system is a computer system able to converse with humans via everyday spoken language. Such systems are expected to play an ever-increasing role in our interactions with technology. They have a wide spectrum of applications, ranging from voice-enabled mobile applications to navigation assistants, smart home environments, tutoring systems, and (in a not-too-distant future) service robots assisting us in our daily chores.

Figure 1.1 depicts an example of interaction between a human user and a spoken dialogue system. When the user starts talking, the system extracts the corresponding speech signal through a microphone. The speech signal is then processed to analyse its content. Once this analysis is completed, the system must then determine how to react. In this case, the system decides to greet back the user and selects the words to express it ("*good morning, sir*"). The final step is then to synthesise these words through an artificial voice, which closes the loop.[1]

1.1 Motivation

Although spoken dialogue systems can greatly enhance the user interaction experience in many of today's technologies, their practical development can be a demanding enterprise. Speech is indeed much more complex than other user interfaces such as keyboards or touch screens.

[1] Needless to say, the schema hides a great deal of internal complexity. The next chapter describes in more detail the software architectures used to design practical spoken dialogue systems.

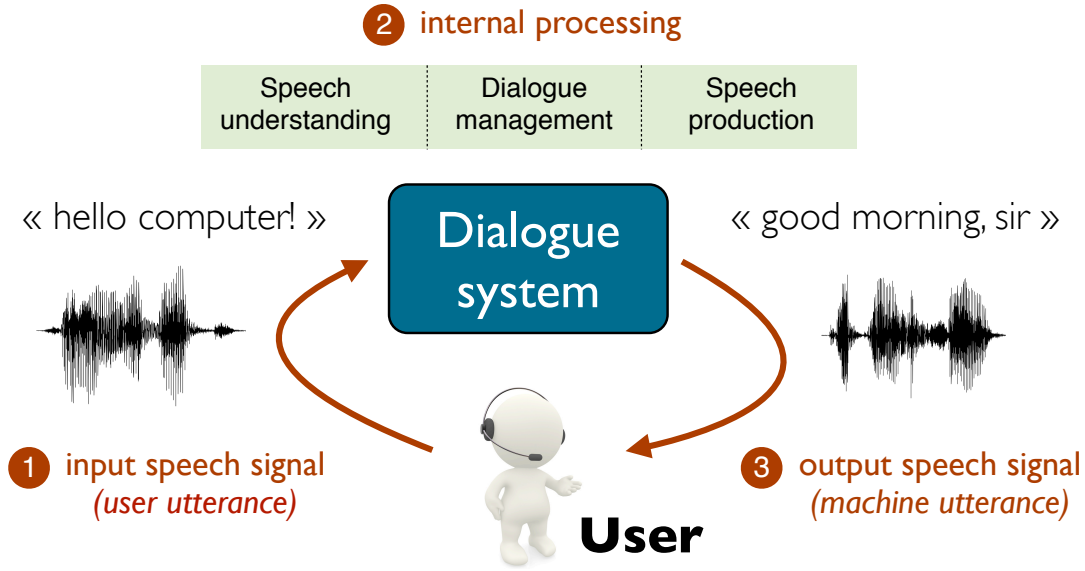


Figure 1.1: Schematic view of a spoken dialogue system.

The present thesis concentrates on the problem of *dialogue management*. Dialogue management is a central component in spoken dialogue systems and lies the intersection between speech understanding and production. It serves a double role. The first function of dialogue management is to maintain a representation of the current dialogue state. This representation reflects the system knowledge of the current conversational situation, and often include multiple features related to the dialogue history, the external context, and the status of the tasks to perform. This dialogue state is regularly updated with new information in the form of new user utterances or perceived changes in the context. The second function of dialogue management is to make decisions. Based on the current state of the interaction, dialogue management must decide which actions to undertake. These actions are often communicative in nature (e.g. uttering a sentence), but can also pertain to physical actions to execute (e.g. grasping an object).

Dialogue management is therefore responsible for controlling the flow of the interaction, by (1) interpreting the user intentions in their context and (2) selecting which actions to perform. In the example from Figure 1.1, this step corresponds to the decision of responding to the user utterance "*hello computer!*" with another greeting action, "*good morning, sir*".

Along with speech recognition, dialogue management is arguably one of the most difficult processing task in spoken dialogue systems. This difficulty stems from two defining characteristics of verbal interactions:

1. Verbal interactions are *complex*. Taking part in a dialogue requires tracking a multitude of factors, such as the interaction history, the hypothesised goals and preferences of the dialogue participants, and the external situation. These factors depend on one another through multiple relations straddling the linguistic and extra-linguistic boundaries. Selecting the action that is most appropriate in a particular situation is thus a difficult decision problem.

2. Verbal interactions are also crippled with *uncertainties*. In order to make sense of a given dialogue, a conversational agent must face numerous sources of uncertainty, including error-prone speech recognition, lexical, syntactic and referential ambiguities, partially observable

environments, and unpredictable interaction dynamics.

The combination of these two properties forms an explosive mix. In order to make sense of the interaction and act appropriately, the dialogue system must resort to sophisticated reasoning in order to interpret the user intentions in their context and plan the best course of action. And it must do so under high levels of noise and uncertainty, where many pieces of information can be erroneous, missing, ambiguous, or fragmentary. This task is defined in the field of artificial intelligence as *sequential decision-making under uncertainty*, and is known to be a particularly difficult computational problem, especially for complex domains such as dialogue. Decision-making and action execution must also occur in real-time, since dialogue is by nature a real-time process.

Research on dialogue management can be divided into two main lines of investigation that reflect their focus on either of the two challenges we just mentioned.

On the one hand, structural complexity is often dealt with conceptual tools borrowed from formal logic and classical planning. These approaches provide principled methods for the interpretation and generation of dialogue moves through logical reasoning on the basis of a formal representation of the mental states of the dialogue participants (including their shared knowledge). Based on such representation, dialogue is then framed as a collaborative activity in which the dialogue participants act together to coordinate their actions, maintain a shared conversational context, resolve open issues and satisfy social obligations (Larsson, 2002; Jokinen, 2009; Ginzburg, 2012). These approaches can yield detailed analyses of various conversational behaviours, but they generally assume complete observability of the dialogue state and provide only a limited account of errors and uncertainties. In addition, they require the knowledge base on which the inference is grounded to be completely specified in advance by domain experts. Their deployment in practical applications is therefore non trivial.

On the other hand, the problem of uncertainty is usually addressed by probabilistic modelling techniques (Roy et al., 2000; Frampton and Lemon, 2009; Young et al., 2010). The state of the dialogue is here represented as a probability distribution over possible worlds. This distribution represents the system's current knowledge of the interaction and is regularly updated as new observations are collected. These probabilistic models provide an explicit account for the various uncertainties that can arise during the interaction. They also enable the dialogue behaviour to be automatically optimised in a data-driven manner instead of relying on hand-crafted mechanisms. Dialogue strategies can therefore be adapted to new environments or users without having to be reprogrammed. However, these models typically depend on large amounts of training data to estimate their parameters – a requirement that is hard to satisfy for most dialogue domains. In addition, the probabilistic models are usually limited to a handful of state variables and are difficult to scale to domains featuring rich conversational contexts.

The work described in this thesis aims at reconciling these two strands of research through a new, hybrid framework to dialogue modelling and control.

1.2 Contributions

The present thesis details an original approach to dialogue management based on *structured probabilistic modelling*. The overarching objective of this work is to design probabilistic models of dialogue that are scalable to rich conversational domains, yet only require limited amounts of training

data to estimate their parameters.

An extensive body of work in the machine learning and decision-theoretic planning literature shows how to confront this issue by relying on more expressive representations, able to capture relevant aspects of the problem *structure* in a compact manner. By taking advantage of hierarchical or relational abstractions, system developers can leverage their domain knowledge to yield probabilistic models which are both easier to learn (due to a reduced number of parameters) and more efficient to use (since the structure can be exploited by the inference algorithm).

This thesis demonstrates how to translate these insights in dialogue modelling.

Probabilistic graphical models (Koller and Friedman, 2009) form the theoretical foundations for a large part of our work. Graphical models provide a generic, principled framework for representing and reasoning over complex probabilistic problems. They also come with well-defined data structures and efficient general-purpose algorithms for model estimation and inference. As shown by e.g. Thomson and Young (2010), the dialogue state can be elegantly represented as a Bayesian network (a well-known type of directed graphical model) factored in a set of state variables describing various aspects of the conversational situation. The dialogue state is graphically depicted as a directed acyclic graph where the nodes correspond to particular variables and the edges are conditional dependencies between variables. To exploit such representation for decision-making purposes, the dialogue state can be extended with action and utility nodes that describe the utility for the agent of performing particular actions in a given situation.

The statistical estimation of such complex probabilistic structures is however a non-trivial task owing to the large number of variables and dependencies involved. The main novelty of our approach is the idea of representing the model distributions in a structured manner through the use of *probabilistic rules*. These rules encode the conditional distributions between variables in terms of structured mappings associating particular conditions defined on a set of input variables to probabilistic effects defined on a set of output variables. Utility distributions can also be defined in a similar manner. The conditions and effects are expressed as logical formulae and can make use of a restricted form of universal quantification in order to handle relational domains. As new information becomes available to the dialogue manager, the Bayesian network representing the current dialogue state is updated by instantiating the rules in the form of new nodes that mediate between input and output variables. Probabilistic rules are therefore employed as high-level templates for the generation of a classical probabilistic model.

The resulting modelling framework offers two major benefits. Most importantly, the reliance on more expressive representations can drastically reduce the number of parameters associated with the models. Instead of being encoded through traditional probability tables, the conditional distributions between states variables are expressed through high-level rules that capture conditional dependences with a compact set of parameters (one for each possible effect). As a consequence, these models are much easier to learn and generalise to unseen data. In addition, the framework enables expert knowledge to be directly integrated in the probabilistic dialogue models. System developers can therefore exploit powerful abstractions to encode their prior knowledge of the dialogue domain in the form of pragmatic rules or task-specific constraints. While the usefulness of such expert information has long been recognised, its use has most often be reduced to a mere external filter for classical statistical models (Heeman, 2007; Williams, 2008b). By contrast, our approach incorporates such knowledge in the very structure of the statistical model.

We conducted several experiments to assess the validity of our approach in three distinct learn-

ing scenarios:

1. The first experiment focused on the problem of estimating the utilities of various system actions given a small data set collected from Wizard-of-Oz interactions.[2] Based on dialogue models encoded with probabilistic rules, the utilities of the different actions were learned through imitation learning. We were able to show that the rule structure enabled the learning algorithm to converge faster and with better generalisation performance than unstructured models. This work was originally presented in (Lison, 2012d).

2. The second experiment extended the above approach to reinforcement learning. The goal of this study was to estimate the transition model of the domain from interactions with a user simulator. We compared the relative learning performance of two modelling approaches: one relying on unstructured distributions, and one based on probabilistic rules. The empirical results demonstrated the benefits of capturing the domain structure with probabilistic rules. The results were first published in (Lison, 2013).

3. Finally, the third experiment was designed to evaluate the approach through live interactions with real users. to be completed

An additional contribution of this thesis is the release of a software toolkit that implements all the data structures and algorithms presented in this work. The toolkit is called `openDial` and is freely available under an open source licence.[3] The purpose of the toolkit is to enable system developers to design and evaluate dialogue systems based on probabilistic rules. All domain-specific knowledge is declaratively specified in the rules for the domain. The system architecture is therefore reduced to a small set of core algorithms for accessing and updating the dialogue state (Lison, 2012a). This architectural design makes the toolkit fully generic and domain-independent. The `openDial` toolkit comes with a user interface allowing developers to interactively test their system and visualise how the internal dialogue state is evolving over time.

We carried out all the experiments described in this thesis in a particular application domain, namely *human–robot interaction* (HRI). The choice of this application domain as a test bed for our framework was motivated by two factors. The first factor is the presence of a complex situated environment in which the agent must complete its tasks. This dialogue context is highly dynamic –

[2] A Wizard-of-Oz interaction is an experimental procedure borrowed from the field of human-computer interaction (Dahlbäck et al., 1993). In a Wizard-of-Oz experiment, the subjects are asked to interact with a computer system which has all the appearances of reality, but is actually remotely controlled by an (unseen) human agent operating behind the curtains. Wizard-of-Oz studies are often conducted to provide the system designers with interaction data from real users before the system is fully implemented. The term is a cultural reference from the 1939 film "The Wizard of Oz" (based on a book by Frank Baum), where an illusionist impersonates a powerful wizard by controlling an intimidating display from behind a curtain.

[3] The toolkit can be downloaded at `http://opendial.googlecode.com`. not ready yet

the physical position of objects and persons may for instance change in the course of the interaction, and the system tasks are regularly altered as a result of the coordinated actions of the robot and human user. The second factor relates to the occurrence of multiple sources of uncertainty caused by e.g. imperfect sensory devices, unreliable motors, and failure-prone speech recognition.[4] This combination of a rich conversational context and high levels of uncertainty is precisely the focus of our thesis work and designates human-robot interaction as an ideal test bed to evaluate the performance our modelling approach in real settings.

The Nao robot from Aldebaran Robotics was used as a development and testing platform in all our experiments.[5] An example of interaction with the robot is shown in Figure 5.6. Most of our experiments involved the Nao robot interacting with a human user in a shared visual environment featuring a few basic objects that can be both perceived and grasped by the robot. The interactions typically revolved around the completion of a few simple tasks such as moving an object from one position to another under the supervision of the human user. Chapters 5–8 provide a detailed description of the interaction scenarios, data collection and evaluation setups followed for the experiments conducted as part of our thesis work.

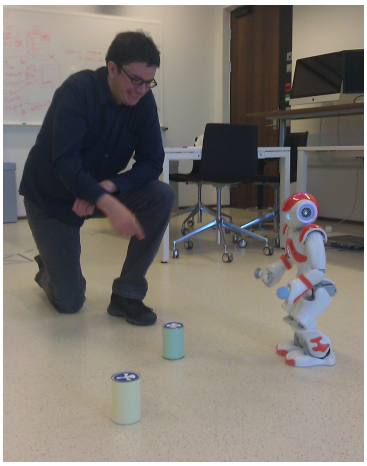


Figure 1.2: Human user interacting with the Nao robot.

1.3 Outline of the thesis

We provide here a brief outline of the thesis structure, chapter by chapter.

Chapter 2: Background

This chapter introduces the fundamental concepts and methods used throughout this thesis. We start with an overview of some of the core linguistic properties of dialogue and describe key notions such as turn-taking, dialogue acts and grounding. We then describe the software architectures used to design spoken dialogue systems and the role of each component within them. We also mention a range of important applications for spoken dialogue systems. Finally, we survey the various approaches that have been put forward in the research literature to address the dialogue management problem, including both hand-crafted and statistical approaches.

Chapter 3: Probabilistic modelling of dialogue

The chapter starts by reviewing the core notions of directed graphical models, which constitute the formal basis for our framework. We define how Bayesian networks are constructed,

[4] For practical reasons, the microphones are often placed on the robot itself, at a significant distance from the speaker. This distance between source and receiver is a major degradation factor in speech recognition (Wölfel and McDonough, 2009). Moreover, the microphones are also adjacent to a number of mechanical motors which may disturb the sound signal and lead to spurious detections.

[5] cf. http://www.aldebaran-robotics.com.

and show how they can be augmented to capture temporal sequences and decision-theoretic problems. We also briefly describe the most important methods for learning and inference based on such models. We then move to the field of reinforcement learning and spell out its most central elements, such a Markov Decision Processes, value functions and policies. We also examine how reinforcement learning methods can be extended to partially observable settings, since dialogue is a prototypical example of decision-making under partial observability. Finally, the last section translate these concepts to the field of dialogue management, and discusses both supervised and reinforcement learning approaches to the optimisation of dialogue policies.

Chapter 4: Probabilistic rules

This chapter lays down the central concepts and algorithms of our own modelling approach to dialogue management. We define what probabilistic rules are and how they are internally structured through conditions and effects. We describe two main types of rules, used to respectively encode probability and utility distribution. We then explain how the rules are practically instantiated in the Bayesian Network representing the dialogue state, as well as the processing workflow that is followed to update the dialogue state and perform action selection. The chapter also addresses some advanced modelling questions, and concludes by comparing our framework to previous work.

Chapter 5: Learning from Wizard-of-Oz data

This chapter shows how the parameters attached to probabilistic rules can be automatically learned from training data, in a supervised learning fashion. The algorithm used for estimating the rule parameters is grounded in Bayesian learning techniques. To validate our approach, we detail an experiment on a statistical estimation task based on Wizard-of-Oz data collected in a human–robot interaction domain. The experiment illustrates the benefits of probabilistic rules compared to unstructured distributions.

Chapter 6: Learning from interactions

Chapter 6 extends parameter estimation to a reinforcement learning context. We show how the parameters of rule-structured dialogue models can be efficiently learned from observations collected during the interaction itself, without having access to any gold standard annotations. The learning procedure follows a model-based Bayesian reinforcement learning approach. Finally, we report the results of an experiment carried out with a user simulator. The experiment concentrated on the estimation of the transition model for a human-robot interaction domain, and evaluated the relative performance of a model structured with probabilistic rules compared to a plain probabilistic model.

Chapter 7: Implementation

Chapter 7 uncovers how the various algorithms and data structures presented in this thesis are technically integrated in the system architecture. We explain how the `openDial` toolkit is internally organised in distinct modules and present the graphical user interface used to

monitor and visualise the evolution of the dialogue state over the course of the interaction. We also describe how dialogue domains and their parameters are practically specified in a generic XML format and describe the implementation and performance tuning of the algorithms used for probabilistic inference and online planning. Finally, we briefly compare the openDial architecture to related software frameworks.

Chapter 8: User evaluation

This chapter presents a user evaluation of our approach in a HRI domain. XXX

Chapter 9: Concluding remarks

The final chapter concludes this dissertation with a summary of the presented research contributions, followed by an outline of future work.

Chapter 2

Background

We introduce in this chapter the most important concepts and methods employed in the field of spoken dialogue systems, with special emphasis on dialogue management. We start by reviewing some key linguistic concepts that are particularly relevant for our work: turn-taking, dialogue acts and grounding. A proper understanding of these aspects is indeed a prerequisite for the design of conversationally competent dialogue systems. After this linguistic overview, we move to a more technical discussion of the software architectures used to implement practical dialogue systems. These architectures typically comprise multiple processing components, from speech recognition to understanding, dialogue management, output generation and speech synthesis. We briefly describe the role of each component and their positions in the global processing pipeline.

Last but not least, the final section of this background chapter delves into the diverse set of approaches that have been put forward to tackle the dialogue management problem. We first present hand-crafted approaches, starting with finite-state policies and pursuing with more sophisticated methods based on logic- or plan-based reasoning. Finally, we survey the more recently developed statistical approaches to dialogue management that seek to automatically extract dialogue strategies from data, based on supervised and reinforcement learning methods.

2.1 What is spoken dialogue?

We communicate in order to fulfil a wide array of social functions, such as exchanging ideas, recollecting experiences, sustaining relationships, or collaborating with others to accomplish shared goals. These communication skills are developed in early childhood, and our cognitive abilities are in many ways shaped and amplified by this disposition for verbal interaction.

One of the most important property of dialogue is that it is fundamentally a *collaborative activity* (emphasis on both words). It is, first of all, an *activity*, which means that is is (1) motivated by the desire to achieve specific (practical or social) goals; (2) subject to costs to minimise (the communication effort); and (3) composed of a temporal sequence of basic actions. Furthermore, if we abstract from so-called "internal dialogues" with oneself, dialogue involves per definition at least two participants that must act together to keep the dialogue on track. As shown by a wealth of studies in psychology and linguistics (Clark and Schaefer, 1989; Allwood et al., 1992; Clark, 1996; Garrod and Pickering, 2004; Tomasello et al., 2005), human conversations are characterised by a high degree of *collaboration* between interlocutors. The individuals participating in a dialogue routinely collaborate in order to coordinate their contributions and ensure mutual understanding,

thereby making the interaction more efficient. This collaboration is done mostly unconsciously and is part and parcel of the conversational skills we develop as speakers of a given language.

We describe in the next sections four major aspects of this collaborative activity:

1. The dialogue participants take *turns* in a conversation.
2. These turns are structured into basic communicative units called *dialogue acts*.
3. The interpretation of these dialogue acts is subordinated to the *conversational context* in which they are uttered.
4. The participants continuously provide *grounding signals* to each other in order to indicate how they understand (or fail to understand) each other's contributions.

2.1.1 Turn-taking

Turn-taking is one of the most basic (yet often neglected) aspect of spoken dialogue. The physical constraints of the communication channel impose that participants take turns in order to speak. Turn-taking is essentially a resource allocation problem. In this case, the resource to allocate is called the *conversational floor*, and social conventions dictate how the dialogue participants are to take and release their turns.

The field of *conversation analysis* studies what these conventions are and how they combine to shape conversational behaviours. Human conversations are indeed remarkably efficient at turn-taking. Empirical cross-linguistic studies have shown that the average transition time between turns revolves around 250 ms. (Stivers et al., 2009).[1] In addition, most of the utterances do not overlap: Levinson (1983) indicates that less than 5 % of the speech stream contains some form of overlap in spontaneous conversations.

A wide variety of cues are used to detect turn boundaries, such as silence, hesitation markers, syntax (complete grammatical unit), intonation (rising or falling pitch) and body language, as described by Duncan (1972). These cues can occur jointly or in isolation. Upon reaching a turn boundary, a set of social conventions governs who is allowed to take the turn. The current speaker can explicitly select the next person to take the turn, for instance when greeting someone or asking a directed question (Sacks et al., 1974). This selection can also occur via other mechanisms such as gaze. When no such selection is indicated, other participants are allowed to take the turn. Alternatively, the current speaker can continue to hold the floor until the next boundary.

Turn-taking is closely related to the notion of *initiative* in human–computer interaction. The vast majority of dialogue systems currently deployed are either system-initiated or user-initiated. In a system-initiated dialogue, the dialogue system has full control on how the interaction is unfolding – i.e. the system is asking all the questions and waiting for the user responses. A user-initiated dialogue is the exact opposite: in such settings, the user is assumed to lead the interaction and request information from the system. The most complex – but also most natural – interaction style is the mixed-initiative, where both the user and the dialogue system are allowed to take the initiative at any time and decide to either provide or solicit information whenever they see fit (Horvitz, 1999).

[1] Interestingly, this duration is shorter than the time required for a human speaker to plan the motor routines associated with the physical act of speaking. This means that the next speaker must start planning his utterance before the current turn is complete, and predict when a potential turn boundary is likely to appear.

The turn-taking behaviour of most current-day dialogue systems remains quite rudimentary. The most common method to detect the end of a user turn is to wait for a silence longer than a manually fixed threshold, typically ranging between ½ and 1.0 second. Some system architectures also include routines for handling barge-ins – that is, user interruptions – (Ström and Seneff, 2000), while others simply ignore them altogether. Turn-taking has recently become a focus of research in its own right in the dialogue system literature (Raux and Eskenazi, 2009; Gravano and Hirschberg, 2011), in an effort to break away from the ping-pong interaction style that characterises most current dialogue interfaces.

2.1.2 Dialogue acts

Each turn is constituted of one or more utterances. As argued by Austin (1962) and Searle (1969), utterances are nearly always purposeful: they have specific goals and are intended to provoke a specific psychological effect on the listener(s). They are therefore best described as actions rather than abstract statements about the world. The notion of dialogue act embodies precisely this idea.[2] Bunt (1996) defines a dialogue act as a "functional unit of a dialogue used by the speaker to change the context".

In his seminal work on the philosophy of language, Searle (1979) established a taxonomy of speech acts divided in five central categories:

Assertives: Committing the speaker to the truth of a proposition.
Examples: "*I swear I saw him on the crime scene.*", "*I bought more coffee.*"

Directives: Attempts by the speaker to get the addressee to do something.
Examples: "*Clean your room!*", "*Could you post this for me?*"

Commissives: Committing the speaker to some future course of action.
Examples: "*I will deliver this review before Monday.*", "*I promise to work on this.*"

Expressives: Expressing the psychological state of the speaker about a state of affairs.
Examples: "*I am so happy for you!*", "*Apologies for being late.*"

Declaratives: Bringing about a different state of the world by the utterance.
Examples: "*You're fired.*", "*We decided to let you pass this exam.*"

Modern taxonomies of dialogue acts are significantly more detailed than the one introduced by Searle. They also provide detailed accounts of various dialogue-level phenomena such as grounding (cf. next section) that were absent from Searle's analysis. The most well-known annotation scheme is DAMSL (Dialogue Act Markup in Several Layers) and was initially formalised by Core and Allen (1997). DAMSL defines a rich, multi-layered annotation scheme for dialogue acts that is both domain- and task- independent. A modified version of this scheme was applied to annotate

[2] Dialogue acts have gone through multiple names over time, owing to the diverse range of research fields that have studied them, from philosophy to descriptive and computational linguistics. As listed in McTear (2004), alternative denominations include speech acts (Searle, 1969), communicative acts (Allwood, 1976), conversation acts (Traum and Hinkelman, 1992), conversational moves (Sinclair and Coulthard, 1975), and dialogue moves (Larsson et al., 1999).

the Switchboard corpus[3] based on a set of 42 distinct dialogue acts (Jurafsky et al., 1997), including greeting and closing actions, acknowledgements, clarification requests, self-talk, responses, and many more. An interesting aspect of DAMSL is the use of two complementary dimensions in the markup: the *forward-looking functions*, which are the traditional speech acts in Searle's sense (assertions, directives, information requests, etc.) and the *backward-looking functions* that respond back to a previous dialogue act and can signal agreement, understanding, or provide answers. Both backward- and forward-looking functions can be present in the same utterance.

Determining the dialogue act corresponding to a given utterance is a non-trivial operation. The type of utterance only gives a partial indication of the underlying dialogue act – a question can for instance express a directive ("*Could you post this for me?*"). In order to accurately classify a dialogue act, a variety of linguistic factors must be taken into account, such as prosody, lexical, syntactic and semantic features, and the preceding dialogue history (Jurafsky et al., 1998; Shriberg et al., 1998; Stolcke et al., 2000; Keizer and op den Akker, 2007).

2.1.3 Interpretation of dialogue acts

Dialogue acts are strongly contextual in nature: their precise meaning can often only be comprehended within the particular conversational context in which they appear. The successful interpretation of dialogue acts must therefore venture beyond the boundaries of the isolated utterance. We briefly review here three striking aspects of this dependence on context.

Implicatures

As shown by Grice (1989), an important part of the semantics of dialogue acts is not explicitly stated but rather implied from the context. Consider the following constructed example:

A: Is William working today?
B: He has a cold.

In order to retrieve the "suggested" meaning behind B's utterance – namely, that William is probably not working –, one needs to assume that B is cooperative and that his response is therefore relevant to A's question. If an utterance initially seems to deliberately violate this principle, the listener must search for additional hypotheses required to make sense of the dialogue act. Grice (1989) formalised these ideas in terms of a cooperative principle composed of four conversational maxims that are assumed to hold in a natural conversation: the maxim of quality ("be truthful"), the maxim of quantity ("be exactly as informative as required"), the maxim of relation ("be relevant") , and the maxim of manner ("be relevant"). These notions have been further developed by various theorists such as Wilson and Sperber (2002) and Horn and Ward (2008). A computational account of these implicatures (and application to dialogue systems) is provided by Benotti (2010).

Non-sentential utterances

Non-sentential (also called elliptical) utterances are linguistic constructions that lack an overt predicate. They include expressions such as "*where?*", "*at 8 o'clock*", "*a bit less, thanks*" and "*brilliant!*".

[3]The Switchboard corpus is a corpus of spontaneous telephone conversations collected in the early 1990's. It includes about 2430 conversations averaging 6 minutes in length; totalling over 240 hours of recorded speech with native speakers of American English (Godfrey et al., 1992).

Their interpretation generally requires access to the recent dialogue history to recover their intended meaning. This can lead to ambiguities in the resolution, as illustrated in these examples modified from Fernández et al. (2007):

- A: "When do they open the new station?" → B: "Tomorrow" (*short answer*)
- A: "They open the station today" → B: "Tomorrow" (*correction*)
- A: "They open the station tomorrow" → B: "Tomorrow" (*acknowledgement*)

Various accounts of non-sentential utterances have been proposed, based on e.g. discourse coherence (Schlangen and Lascarides, 2003) or interaction-oriented semantics (Fernández, 2006; Ginzburg, 2012). Machine learning approaches have also been developed (Schlangen, 2005; Fernández et al., 2007).

Referring expressions

Finally, dialogue acts are replete with linguistic expressions that refer to some aspect of the conversational context. These references can be either deictic or anaphoric.

A deictic marker is a reference to an entity that is determined by the context of enunciation. Examples of such markers are "*here*" (spatial reference), "*yesterday*" (temporal reference), "*this mug*" (demonstrative), "*you*" (reference to a person), or even pointing gestures. By their very definitions, deictic markers refer to different realities depending on the situation in which they are used: a "*here*" uttered in a classroom differs from a "*here*" uttered in the countryside.

In addition, dialogue can also include anaphoric expressions – that is, expressions that refer to an element that has been previously mentioned through the history of the dialogue. An simple example of such anaphoric expression can be seen in the question-answer pair "*Is William working today?*" → "*He has a cold*", where the pronoun "*he*" must be resolved to "*William*".

The appropriate processing of deictic and anaphoric expressions is an important question in dialogue systems, and pertains both to the interpretation and production process. Multiple approaches have been pursued, relying on symbolic (Eckert and Strube, 2000) or statistical techniques (Strube and Müller, 2003; Stent and Bangalore, 2010). Researchers have also investigated the integration of salience measures (Kelleher and Van Genabith, 2004), multimodal cues (Frampton et al., 2009; Chen et al., 2011), the processing of spatial referring expressions (Zender et al., 2009) and the incrementality of the resolution process (Schlangen et al., 2009; Poesio and Rieser, 2011).

2.1.4 Grounding

Dialogue acts are executed as part of a larger collaborative activity that requires the active coordination of all conversational partners, i.e. speaker(s) as well as hearer(s). This coordination takes place at various levels. The first and most visible level is the content of the conversational activity. The partners must ensure mutual understanding of each other's contribution, to control that they remain "on the same page". In addition, they also coordinate the process by which the conversational activity moves forward – by signalling that they are attending to the person who currently holds the conversational floor and acknowledging her/his contributions to the dialogue.

As an illustration, consider this short excerpt from a conversation transcribed in the British National Corpus (Burnard, 2000) :

KATHLEEN : How come they can take time off yet you can't?
STEVE : He's been there longer than me.
KATHLEEN : Oh.
STEVE : I can, I might have two holidays now, two days' holiday. ...
KATHLEEN : Well ... I don't get that, me.
STEVE : What?
KATHLEEN : All these two days' holiday and this, you've had Christmas.
STEVE : You get two point summat[4] days per month worked
KATHLEEN : Oh so you should've got them for January? ...
STEVE : right?
KATHLEEN : Yeah.
STEVE : And I worked three month before Christmas so I got six point summat days
KATHLEEN : For Christmas.
STEVE : so then I had all Christmas off.
KATHLEEN : Oh!
Yeah I get it now.
... I thought you got Christmas off like we got Christmas off.
STEVE : No.
You gotta earn them. ...

(http://www.phon.ox.ac.uk/SpokenBNCdata/KCX.html)

We can observe in this short dialogue that the interlocutors constantly rely on the *common ground* of the interaction to move the dialogue forward. They regularly check what pieces of information are mutually known and understood (e.g. "*right?*"). They also make use of a variety of signals to indicate when things are properly grounded ("*oh*", "*yeah*", "*I get it*") and when they are not ("*I don't get that*", "*what?*"). The common ground progressively expands as the dialogue unfolds – for instance, the system of holiday entitlement is not initially part of the shared knowledge for both speakers at the onset of the conversation, but becomes so towards the end.

The common ground is defined as the collection of shared knowledge, beliefs and assumptions that is established during an interaction.[5] Each dialogue act is built upon the current common ground and participates in its gradual expansion and refinement. This process is called *grounding*. A variety of feedback mechanisms can be used to this effect. As described by Clark and Schaefer (1989), positive evidence of understanding can be expressed via cues such as:

Continued attention: The hearer shows that he/she continues to attend to the speaker.

Relevant next contribution: The hearer produces a relevant follow-up, as in the answer "*He's been there longer than me*" following the question that precedes it.

[4] "Summat" is slang for "something" in the Yorkshire region.

[5] An information that is part of the common ground for a given group is more than simply known by every member of the group. All group members must also be aware that the information is shared and known by the other members. Formally speaking, a proposition p is part of the common knowledge for a group of agents G when all the agents in G know p, and they also all know that they all know p, and they all know that they all know that they all know p, *ad infinitum*. This definition can be formalised using the mathematical apparatus of set theory or epistemic logic (Meyer and Van Der Hoek, 2004).

Acknowledgement: The hearer nods or utters a backchannel such as "*mm*", "*uh-uh*", "*yeah*", or an assessment such as "*I see*", "*great*", "*I get it now*".

Demonstration: The hearer demonstrates evidence of understanding by reformulating or completing the speaker utterance.

Display: The hearer reuses part of the previous utterance.

Communication problems can also occur, owing to e.g. misheard or misunderstood utterances. The hearer should in this case provide negative feedback to signal trouble in understanding. A large panel of clarification and repair strategies are available to recover from these communicative failures. These strategies include backchannels ("*mm?*"), confirmations ("*Do you mean that...?*"), requests for disambiguations, invitations to repeat, and tentative corrections.

All in all, these positive and negative signals enable the dialogue participants to dynamically synchronise what the speaker intends to express and what the hearers actually understand. This grounding process operates mostly automatically, without deliberate effort. It is closely related to the concept of interactive alignment that has recently been articulated by Garrod and Pickering (2004, 2009). Humans show a clear tendency to (unconsciously) imitate their conversational partners. In particular, they automatically align their choice of words, a phenomenon called lexical entrainment (Brennan and Clark, 1996). But alignment also occurs on several other levels such as grammatical constructions (Branigan et al., 2000), pronunciation (Pardo, 2006), accents and speech rate (Giles et al., 1991), and even gestures and facial expressions (Bavelas et al., 1986).

A proper treatment of grounding is critical for the development of conversational interfaces. As already mentioned in the introductory chapter, comprehension errors are indeed ubiquitous in spoken dialogue systems. The potential sources of misunderstandings are abundant, from error-prone speech recognition to out-of-domain utterances, unresolved ambiguities, and unexpected user behaviour. Appropriate grounding strategies are crucial to address these pitfalls. Grounding for dialogue systems is an active area of research and important advances have been made regarding the formalisation of rich computational models of grounding (Traum, 1994; Matheson et al., 2000), the generation of clarification requests (Purver, 2004; Rieser and Moore, 2005), the design of human-inspired error handling strategies (Skantze, 2007), the integration of non-verbal cues such as gaze, head nods and attentional focus (Nakano et al., 2003) and the development of incremental grounding mechanisms (Visser et al., 2012).

2.2 Spoken dialogue systems

After reviewing some of the core properties of human dialogues, we now discuss how to develop practical computer systems that aim to emulate such type of conversational behaviour. In the introduction chapter, Figure 1.1 represented a dialogue system as a black box taking speech inputs from the user and generating spoken responses. Real-world dialogue systems have however a complex internal structure, as we detail in the next pages.

2.2.1 Architectures

Spoken dialogue systems (SDS) often take the form of complex software architectures that encompass a wide range of interconnected components. These components are dedicated to various tasks

related to speech processing, understanding, reasoning and decision-making. These tasks can be grouped into five major components:

1. *Speech recognition*, in charge of mapping the raw speech signal to a set of recognition hypotheses for the user utterance(s).

2. *Natural language understanding*, in charge of mapping the recognition hypotheses to high-level semantic representations of the dialogue act performed by the user.

3. *Dialogue management*, in charge of interpreting the purpose of the dialogue act in the larger conversational context and deciding what communicative action to perform (if any).

4. *Natural language generation*, in charge of finding the best linguistic (and extra-linguistic) realization for the selected communicative action.

5. And finally, *text-to-speech synthesis*, in charge of synthesizing an audio signal out of the generated utterance.

Figure 2.1 shows the flow of information for a prototypical spoken dialogue system. Many systems rely on additional middleware to act as a "software glue" between the components and handle the information exchange and scheduling of modules (Turunen, 2004; Herzog et al., 2004; Bohus and Rudnicky, 2009a; Schlangen et al., 2010).

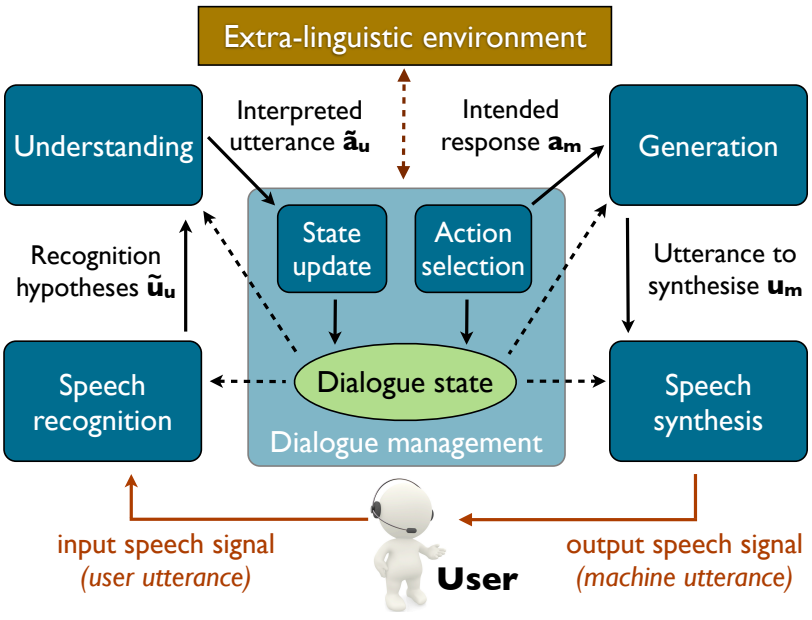


Figure 2.1: Information flow for a typical spoken dialogue system. The solid lines denote necessary input and outputs while the dotted lines represent optional contextual information.

Spoken dialogue systems can use other modalities than speech. In particular, additional communication channels such as touch, gestures, gaze, and other body movements can be fruitfully exploited. As shown by e.g. Wahlster (2006), multiple modalities can be employed to enrich communication in both directions (understanding and generation). In particular, the system can refine

its understanding of the actual user intentions by fusing information perceived through multiple information channels such as gestures (Stiefelhagen et al., 2004) or gaze (Koller et al., 2012). Non-verbal modalities can also be put to use to enhance how information is presented back to the user and convey additional grounding signals, through e.g. facial expressions and gestures. The use of multiple modalities can notably reduce understanding errors and cognitive load (Oviatt et al., 2004) as well as improve the overall user experience (Jokinen and Hurtig, 2006). For all their advantages, multimodal architectures pose however a number of additional challenges related to timing, synchronisation (Salem et al., 2013) and increased system complexity.

In addition to these non-verbal modalities, many dialogue domains are also grounded in an external context that must be accounted for. This external context might be a physical environment for human-robot interaction, a virtual world for embodied virtual agents, a spatial location for in-car navigation systems, or simply a database of factual knowledge for information systems. Contextual factors of relevance for the application must be continuously monitored by the dialogue system (and updated whenever necessary), as many components depend on the availability of such context model for their internal processing. Furthermore, the agent can often actively influence this context through external actions – for instance, a grasping action will modify the location of the gripped object. This contextual awareness necessitates the integration of additional functionalities for perception and actuation. In human–robot interaction domains, these extra-linguistic modules can notably include subsystems for object and scene recognition, spatial navigation, and various motor routines for locomotion and manipulation (Fritsch et al., 2005; Goodrich and Schultz, 2007; Hawes et al., 2007).

Several types of architectures have been proposed to assemble these components in a unified framework. The simplest approach is to arrange the components sequentially in a pipeline starting from speech recognition and ending with speech synthesis. This approach, although relatively straightforward to develop, suffers from a number of shortcomings, amongst which the rigidity of the information flow and the difficulty of inserting feedback loops between components. Pipelines also offer poor turn-taking capabilities, since the system is unable to react before the pipeline has been fully traversed (Raux and Eskenazi, 2009). More advanced architectures – including the one put forward in this thesis – are based on the notion of *information state* (Larsson and Traum, 2000c; Bos et al., 2003). These approaches are essentially blackboard architectures revolving around a central dialogue state that is read and written by various modules connected to it. These modules monitor the state for relevant changes, in which case they trigger their processing routines and update the state with the result. The main advantages of such architectures are (1) a more flexible information flow, since the modules are allowed to process and update information in any order, and (2) the possibility to define modules that take full advantage of the contextual information encoded in the dialogue state. Figure 2.2 provides a graphical illustration of the difference between pipeline and information-state-based architectures.

Finally, a last aspect of dialogue system architectures that has been subject to recent research pertains to *incremental processing*. Many dialogue architectures must wait for an utterance to be fully pronounced to start its interpretation and decide on subsequent actions. This workflow usually leads to poor reactivity and unnatural conversational behaviours. To address this shortcoming, new architectures have been proposed to integrate incremental processing at various stages of interpretation and decision-making (Schlangen and Skantze, 2009).

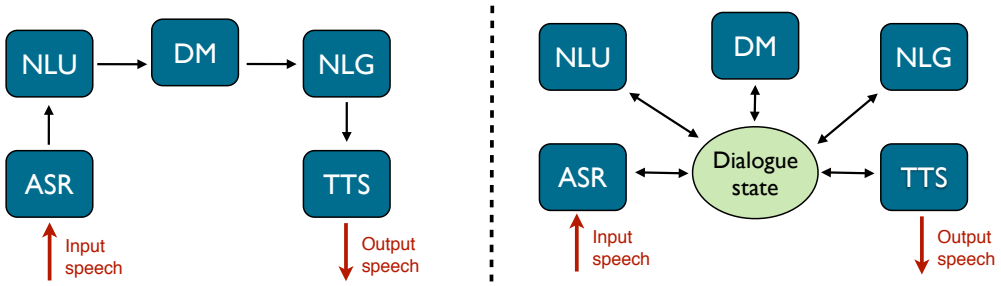


Figure 2.2: Comparison between pipeline (left) and information state (right) system architectures. ASR = *Automatic Speech Recognition*, NLU = *Natural Language Understanding*, DM = *Dialogue Management*, NLG = *Natural Language Generation*, and TTS = *Text-to-Speech Synthesis*.

2.2.2 Components

As explained in the previous section, the components of a dialogue systems can typically be grouped in five major steps. We briefly describe here the role of these components and define their respective inputs and outputs.

Speech recognition

Upon detection of a new speech signal emanating from the user, the first task is to recognise the corresponding utterance. Speech recognition is responsible for converting the raw speech signal from the microphone(s) into a set of hypotheses $\tilde{u}_u$ representing the words uttered by the user. To this end, the speech signal is first converted into a digital format and split into short frames (usually 10 ms). A set of acoustic features is then extracted for each frame using signal processing techniques. Once these acoustic features are extracted, two statistical models are combined to estimate the most likely recognition hypotheses: the *acoustic model* and the *language model*.

The acoustic model defines the observation likelihood of particular acoustic features for a given phone[6], while the language model defines the probability of a given sequence of words. This distinction rests on the formalisation of the speech recognition task as a *Hidden Markov Model* (HMM), where the states represent the sequence of phones, and the observations are the acoustic features.

For the practical development of spoken dialogue systems, the most important element of a speech recogniser is the language model. The language model effectively represents the set of utterances that can be accepted as inputs to the system (and their relative probabilities). The model can be encoded either in the form of a hand-crafted recognition grammar, or via statistical modelling based on a particular corpus of reference. In the latter case, the language model typically takes the form of an N-gram model, often a bi- or tri-gram corrected with appropriate smoothing and back-off techniques (Jelinek, 1997; Chen and Goodman, 1999). It is also often beneficial to dynamically modify the language model at runtime to reflect the changing context and dialogue state. This real-time model adaptation can notably be realised by priming the words or expressions that are most contextually relevant (Gruenstein et al., 2005; Lison, 2010a).

[6] A phone is an individual sound unit of speech. Technically speaking, acoustic models are not defined over entire phones but over sub-segments, typically decomposed into three parts: beginning, middle and end.

The output of the speech recogniser is typically a N-best list (or recognition lattice) representing a set of possible hypotheses for the utterance, together with their relative confidence score or probabilities. Thus, the output of the speech recogniser is a list expressed as:

$$\tilde{u}_u = \langle (\tilde{u}_u^{(1)}, p^{(1)}), (\tilde{u}_u^{(2)}, p^{(2)}), ...(\tilde{u}_u^{(n)}, p^{(n)}) \rangle$$

where $\tilde{u}_u^{(i)}$ represents a specific recognition hypothesis and $p^{(i)}$ its corresponding probability.[7]

Natural language understanding

Once the recognition hypotheses for the utterance have been generated by the speech recogniser, the next task is to extract its semantic content. The goal of natural language understanding (NLU) is to build a representation of the meaning(s) expressed by the form of a given utterance. This task is a notoriously difficult endeavour, due to the combination of various factors. The first difficulty lies in speech recognition errors, with WER (Word Error Rates) often revolving around 20 % for many dialogue applications. The syntactic and semantic analysis of utterances is likewise complicated by the occurrence of sentential fragments, disfluencies of various sorts (e.g. filled pauses, repetitions, corrections) and ambiguities that must be resolved at multiple linguistic levels.

Natural language understanding can be decomposed in a number of steps. Parsing corresponds to the task of extracting the syntactic structure of the utterance and mapping it to a semantic representation. Spoken language parsing can be realised through various techniques, from keyword or concept spotting (Komatani et al., 2001; Zhang et al., 2007) to shallow semantic parsing (Coppola et al., 2009), grammar-based parsing (Van Noord et al., 1999) and statistical parsing (He and Young, 2005). It has been shown useful to apply upstream preprocessing techniques to correct speech recognition errors (Ringger and Allen, 1996) and filter out disfluencies (Johnson and Charniak, 2004). In addition, referring expressions might also need to be resolved (Funakoshi et al., 2012). Finally, the dialogue act associated with the utterance must be determined (Stolcke et al., 2000; Keizer and op den Akker, 2007). De Mori et al. (2008) provides a survey of the various models and techniques used in the field of spoken language understanding.

Given speech recognition hypotheses $\tilde{u}_u$ given as inputs, and possibly a representation of the dialogue history and external context, the task of natural language understanding is to extract a corresponding N-best list of dialogue act hypotheses $\tilde{a}_u$ defined as:

$$\tilde{a}_u = \langle (\tilde{a}_u^{(1)}, p^{(1)}), (\tilde{a}_u^{(2)}, p^{(2)}), ...(\tilde{a}_u^{(n)}, p^{(n)}) \rangle$$

where $\tilde{a}_u^{(i)}$ represents a dialogue act hypothesis, usually represented in a logical form with various predicates and arguments, and $p^{(i)}$ its corresponding probability.

Dialogue management

Dialogue management occupies a central stage in spoken dialogue systems. As already mentioned in the introductory chapter, dialogue management serves a double role. The first task of the dia-

[7] In order to be proper probabilities, the usual axioms $0 \leq p^{(i)} \leq 1$ for all $p^{(i)}$ and $\sum_{i=1}^{n} p^{(i)} = 1$ must be satisfied. It should also be noted that in practice, many speech recognisers only provide raw confidence scores for their hypotheses. Estimating the exact correspondence between these scores and meaningful probabilities is a non-trivial task that has been investigated by e.g. Williams (2008a).

logue manager is to maintain a representation of the current dialogue state and update it as new information becomes available.[8] This dialogue state should encode every information that is of general relevance for the system, such as the recent dialogue history (encoded as a temporally ordered sequence of dialogue acts performed by the dialogue participants), the current conversational floor, the status of the task(s) to fulfil, and various features describing the context of the interaction. Furthermore, the dialogue state can also include information that is indirectly inferred from the individual observations. In particular, many dialogue domains include a variable that explicitly encode the hypothesised user intention. This user intention, although never directly observed, can often be derived from the user inputs through a sequence of reasoning steps. Similarly, the dialogue state can also integrate features that characterise the user profile and her/his preferences. Depending on the theoretical premises chosen for the system, the dialogue state can be either encoded as a fully observable data structure or represent partial observability through the definition of probability distributions on the values of the state variables.

The second task of dialogue management is to take decisions based on this dialogue state. This task is often called *action selection*. The dialogue manager is responsible for selecting the next action to perform by the system, which can be a communicative action (e.g. a piece of information to communicate, a question to task, a grounding signal to convey), an external action (e.g. a physical movement for a robot or a database manipulation for a booking system), a combination of the two, or no action at all. The action selection mechanism can take many forms, ranging from a direct mapping between states and actions to the application of logical rules or the use of offline or online planning techniques.

Dialogue management leads to two distinct outcomes: (1) an updated dialogue state that reflects the observations received as inputs (user dialogue acts, contextual changes etc.), and (2) a selected system action denoted as a_m (the m subscript standing for "machine" to distinguish it from the user act a_u). As for the user dialogue act a_u, the system action a_m is often represented in a logical form composed of predicates and arguments.

Section 2.3 describes in more detail the various approaches and techniques that have been proposed in the literature to formalise the dialogue management process.

Generation

Assuming the selected system action a_m relates to a verbal action, the following step is to find the best linguistic realisation for the abstract communicative goal defined in a_m. As for natural language understanding, a variety of generation techniques are available, from shallow generation strategies based on canned sentences or templates to more more sophisticated approaches based on sentence planning and surface realisation (Stone et al., 2003; Koller and Stone, 2007). More recently, statistical methods have also been pursued in enhance the robustness and user-adaptivity of the generation algorithms (Rieser and Lemon, 2010a; Dethlefs and Cuayáhuitl, 2011).

The inputs of the generation module are the selected system action a_m and optionally the features defined in the dialogue state s (e.g. the user model and the external context). Given this information, the generation module will produce a corresponding user utterance denoted u_m. In

[8] Some approaches explicitly distinguish between two types of management tasks: task management, responsible for monitoring and advancing the execution of the application objectives, and dialogue management *stricto sensu*, responsible for the more conversational aspects of the interaction. Establishing the boundary between the two types of tasks is however not always trivial.

the case of multimodal systems, the module may also deliver realisations for other modalities than the speech channel, such as gestures or facial expressions.

Speech synthesis

The final step of the processing cycle is to synthesise the utterance in a speech waveform – a process called *text-to-speech synthesis*. This synthesis is performed in two consecutive stages. The utterance is first converted into a phonemic representation. This conversion involves multiple processing steps related to text normalisation, phonetic and prosodic analysis. Once the conversion is completed, the resulting phonemic representation is fed to a synthesiser in charge of producing the actual waveform. This synthesis can either be performed by gluing together pre-recorded units of speech from a speech database (concatenative synthesis) or by generating sounds using explicit acoustic models of the vocal tract (formant and articulatory synthesis). Most current dialogue systems rely on concatenative synthesis, and in particular unit selection (Hunt and Black, 1996).

2.2.3 Applications

Spoken dialogue systems have a wide variety of applications, ranging from academic research prototypes to mature commercial products. The first applications can be found in telephone-based systems for information access and service delivery. A large variety of systems have been developed in this area, for applications as diverse as automated call-routing (Gorin et al., 1997), travel planning (Walker et al., 2001), weather updates (Zue et al., 2000), bus schedule retrieval (Raux et al., 2005) or tourist information (Lemon et al., 2006). The recent emergence of smartphones also led to the development of new voice interfaces for multimodal local search (Ehlen and Johnston, 2013), cross-lingual communication (Xu et al., 2012) and even pedestrian exploration (Janarthanam et al., 2012). Many of these ideas have found their way into commercial products, as evidenced by the success of applications such as Apple's Siri, Nuance's Dragon Go! and Google Now.

Spoken dialogue systems can also be applied in domains where the use of touch interfaces and screens should be avoided because it is impractical or dangerous. This is notably the case for in-car navigation systems (Hansen et al., 2005; Castronovo et al., 2010) where voice interfaces are to be preferred for safety reasons. The recent trends towards ubiquitous computing and "ambient" intelligence in smart home environments also offer promising applications of dialogue system technology (Vipperla et al., 2009; López-Cózar and Callejas, 2010).

Spoken dialogue systems are deployed in increasingly complex and open-ended interaction domains, where the artificial agent is no longer a mere executor of user commands, but increasingly plays the role of a collaborator or intelligent assistant. Conversational interfaces have notably developed in the healthcare sector to monitor – and hopefully improve – the health condition and fitness of patients through interactive dialogues (Bickmore and Giorgino, 2006; Ståhl et al., 2009; Morbini et al., 2012). Substantial research has also been devoted into the development of interactive tutoring assistants in various learning contexts (Chi et al., 2011; Dzikovska et al., 2011; Jan et al., 2011; Traum et al., 2012).

Finally, dialogue systems form an integral part of many robotic systems. Robots are deployed in increasingly social environments, such as homes, offices, schools and hospitals. There is therefore a growing need for robots endowed with communicative abilities. Human-robot interaction is an active area of research and has focused on aspects such as situated dialogue processing (Cantrell

et al., 2010; Kruijff et al., 2010), adaptivity (Doshi and Roy, 2008a), symbol grounding (Roy, 2005; Lemaignan et al., 2012) and multimodal interaction (Stiefelhagen et al., 2004; Salem et al., 2012; Mirnig et al., 2013).

2.3 Dialogue management

Various approaches have been proposed to formalise the dialogue management problem. Common to virtually all approaches to dialogue management is (1) the representation of the agent's knowledge of the current situation in a data structure called the *dialogue state* and (2) the use of a decision mechanism to select the action to perform in each dialogue state. A wide range of strategies have been proposed to represent, update and act upon this dialogue state. We first describe hand-crafted approaches and then move on to the more recently developed statistical methods.

2.3.1 Hand-crafted approaches

Finite-state automata

The simplest approach to dialogue management is based on finite-state automata (FSA). A finite state automaton is defined by a collection of states and directed edges between them. Decision-making is made possible by associating each state with a specific action to execute at that state. Each edge in the automata is labelled with a condition on the user input that, if satisfied, will move the current state from the source of the edge to its target. Figure 2.3 illustrates an example of finite-state automata for a simple, system-initiated interaction that takes user directions. If the user response is different from the five expected inputs, the system will ask the user to repeat until a admissible input is provided. The system will continue to request directions until the "*stop*" command is uttered, in which case a final state is reached.

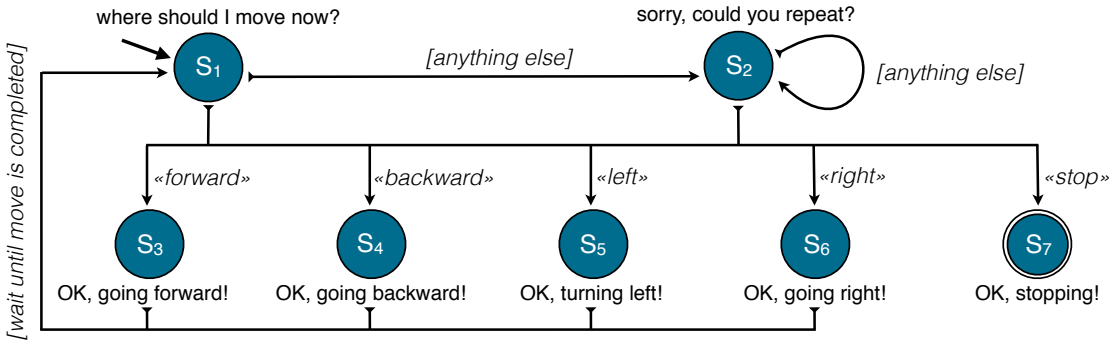


Figure 2.3: Example of finite-state automaton (FSA) for dialogue management, with seven possible states. The starting state for this FSA is s_1 and the (unique) ending state is s_7. The edges $s_3, ...s_7 \rightarrow s_1$ are traversed once the movement is completed, and the two edges $s_1, s_2 \rightarrow s_2$ are traversed for any other user input than the five specified directions.

A finite-state automaton is formally defined as a tuple $\langle \mathcal{S}, \Sigma, \delta, s_0, \mathcal{F} \rangle$, where $\mathcal{S}$ is the set of possible states, Σ the set of inputs that the system can accept (in this case, the user dialogue acts, and possibly external events), $\delta : \mathcal{S} \times \Sigma \rightarrow \mathcal{S}$ the transition function mapping every (state,input) pair to its successor state, s_0 the start state, and $\mathcal{F}$ the set of final states.

Finite-state automata provide a simple and versatile framework for the development of a dialogue manager. Their expressive power is however rather limited, as the dialogue state of a FSA is represented as a single, atomic symbol, and the possible user moves by a finite enumeration of possible transitions allowed for each state. Finite-state automata are therefore difficult to scale to more complex domains where the dialogue state might need to track multiple variables and allow for a large number of user dialogue acts.

Logic- and plan-based approaches

Richer representations of the dialogue state are required to overcome the rigidity of finite-state automata. A popular alternative builds on representations that encode the dialogue state as a frame constituted of a set of slot-value pairs (Seneff and Polifroni, 2000). Frame-based systems start with an empty frame that is gradually filled by the user inputs. After each user move, a set of production rules define what actions to take – typically, a request to elicit a value for a particular slot – based on the current frame. The process continues until all slots are filled, which marks the completion of the dialogue.

Due to their greater expressivity, frame-based systems offer a number of advantages in terms of domain modelling and dialogue control. They remain however difficult to extend to other domains than classical slot-filling applications such as flight booking. The *information state* approach (Larsson and Traum, 2000a) is an attempt to provide a more solid theoretical foundation for dialogue management in rich conversational domains. As already mentioned in Section 2.2.1, information state approaches rely on a blackboard architecture where various modules are attached to a central workspace called the information state. This information state is therefore continuously monitored by the modules integrated the dialogue system, and represent the full contextual knowledge available to the agent. In addition to the usual variables describing the dialogue history and the application task, the information state can also incorporate "mentalistic" entities such as the private and shared beliefs of the conversational agents. The information state can exhibit a rich internal structure encoded as attribute-value matrices (AVMs) or typed records (Cooper, 2012).

Upon reception of a relevant input, the dialogue manager modifies this information state using a collection of update rules. In addition to state-internal operations that modify particular variables of the information state, the update rules are also employed to derive the actions to execute by the agent. Given a collection of rules and a generic strategy to apply them, the dialogue manager can both update its state and select the next action to perform by way of logical inference. This action selection can notably be grounded on the set of open questions raised and not yet answered during the interaction (Larsson, 2002; Ginzburg, 2012).

Plan-based approaches such as the ones developed by Freedman (2000) and Allen et al. (2001) take one step further. These approaches also rely on complex representations of the dialogue state that notably encompass the belief, desires and intentions (BDI) of each agent (Cohen and Perrault, 1979; Allen and Perrault, 1980). But instead of update rules, classical planning is used to update the state and select the next action. In such settings, both the user and the system are assumed to act in pursuit of their long term goals. The interpretation of the user actions is thus cast as a *plan recognition* problem, where the system seeks to derive the belief, desires and intentions that best explain the observed conversational behaviour of the speaker. Similarly, the selection of system actions is derived from the (task-specific) long term objectives of the system. This search for

the best action is an instance of a classical planning task, which can be solved using off-the-shelf planning algorithms. These algorithms require the declaration of a planning domain that specifies the preconditions and effects of every action. Agent-based frameworks such as the Constructive Dialogue Modelling approach developed by Jokinen (2009) follow similar principles, with a particular emphasis on conceptualising dialogue as a collaborative activity grounded in communicative principles of rational and coordinated interaction between agents.

Benefits and limitations of hand-crafted approaches

The primary benefits of hand-crafted approaches to dialogue management lie in their ability to capture rich conversational phenomena and endow the system designer with a fine-grained control over the application behaviour. They have also laid the foundations for substantial advances in the semantic and pragmatic interpretation of dialogue moves (Thomason and Stone, 2006; Ginzburg, 2012), the formalisation of social obligations (Traum and Allen, 1994), the rhetorical structure of dialogue (Asher and Lascarides, 2005), or the use of plan-based reasoning to infer the user intentions (Allen and Perrault, 1980; Litman and Allen, 1987). They nevertheless suffer from two important shortcomings:

1. They generally assume complete observability of the dialogue context and provide only a limited account (if any) of uncertainties. This assumption is unfortunately difficult to reconcile with the imperfections and restricted coverage of speech recognition and understanding.

2. They require the dialogue domain to be specified by hand, either through the definition of an finite-state automaton, a collection of update rules or a set of action schemas for planning. This requirement is hard to satisfy for many domains, since the behaviour of real users is often challenging to anticipate (unsurprisingly, human behaviour can be difficult to predict) and can deviate significantly from the expectations of the system developers.

Statistical approaches, to which we now turn, have been specifically developed to address these two issues.

2.3.2 Statistical approaches

Common to all statistical approaches to dialogue management is the idea of automatically optimising a dialogue policy (that is, a function associating each possible dialogue state to a system action) from interaction data. Starting from this shared premise, statistical approaches vary along multiple dimensions such as the type of learning algorithm, the representation of the dialogue state and policy, and the nature of the data on which to estimate the models. We outline in this section the core concepts of statistical approaches, which will be exposed in a more formal setting in the next chapter.

Supervised learning

The first possible approach is to learn dialogue strategies by imitation based on examples of expert behaviour. This expert behaviour can be recorded through so-called "Wizard-of-Oz" experiments. As already mentioned in the introduction chapter, a Wizard-of-Oz experiment is an interaction in which a human user is asked to interact with a system that is remotely operated by a human

agent (without the user being made aware of this control). A hidden wizard is often preferred to a visible human interlocutor, as people tend to behave differently when they talk to a machine or a human person (Jönsson and Dahlbäck, 1988). One can collect multiple interactions of this type and record the wizard decisions at each point, along with their context. The resulting data set can be fed to a supervised learning algorithm in order to construct a dialogue policy that attempts to imitate the conversational behaviour of the wizard. Learning the dialogue policy is thus seen as a classification problem with states as inputs and actions as outputs. The goal of the learning algorithm is then to construct a classifier that optimises the classification accuracy for the Wizard-of-Oz data set, considering the wizard actions as "gold standards". This classifier can be estimated with any standard machine learning methods (decision trees, logistic regression, etc.).

In a supervised setting, action selection is essentially viewed as a sequence of isolated decision problems. As argued by Levin et al. (2000), this formalisation ignores some important characteristics of conversational behaviour, as dialogue is a dynamic process where the state and action at time t have a direct influence on the resulting state at time $t + 1$. This temporal connection between states is typically lost with classical supervised learning approaches. Furthermore, the state space grows exponentially with the number of state variables, and can therefore reach very large sizes. The training data available from a fixed Wizard-of-Oz corpus will therefore only cover a fraction of the state space for the domain. As a consequence, many states encountered at runtime will have no appropriate training examples on which to ground the action selection. Generalisation and abstraction techniques can however be used to mitigate this problem of data sparsity.

Reinforcement learning

Reinforcement learning (RL) presents an attractive solution to the problem of dialogue policy optimisation. A reinforcement learning problem typically revolves around an *agent* interacting with its environment, typically to perform some practical task. Through its actions, the agent is able to change the state of its environment. After each action, the agent can observe both the new environment state resulting from its actions, as well as a numerical reward encoding the immediate value (positive or negative) of the executed action in relation to the agent's goal. The goal of the learning agent is to find the best action to execute in any given state via a process of trial and error – the best action being characterised as the one that maximise the agent's expected long-term reward.

Reinforcement learning tasks are generally formalised using *Markov Decision Processes* (MDPs), which are defined as tuples $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$ with a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition function T that encodes the probability $P(s'|s, a)$ of reaching state s' after executing action a in state s, and a reward function R that specifies the reward value associated with the execution of action a in state s. Dialogue can be expressed as a Markov Decision Process where the state space corresponds to the possible dialogue states and the actions to the set of (verbal or extra-verbal) actions available to the dialogue agent. The transition function T captures the "dynamics" of the conversation, and indicates how the dialogue state is expected to change as a result of the system actions. Finally, the reward function R expresses the objectives and costs of the application. A common reward function is to assign a high positive value for the successful completion of the task, a high negative value for a failure, and a small negative value for soliciting the user to repeat or clarify her/his intention.

Given a particular MDP problem, the goal of the learning agent is to find a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$

that maps each possible state to the best action to execute at that state. The best action is defined as the action that maximises the *expected return* for the agent. Simply put, the return is the long-term (discounted) accumulation of rewards from the current state up to a given horizon.

Various learning methods have been devised to automatically extract this optimal policy from interaction experience. Due to the large amounts of cycles that are necessary to converge onto an optimal policy, direct interactions with real users are often impossible or highly impractical for many domains. Instead, most recent approaches have relied on the construction of a user simulator able to generate unlimited numbers of interactions on the basis of which the dialogue system can optimise its policy. The user simulator can either be designed by experts or "bootstrapped" from existing datasets or Wizard-of-Oz studies (Pietquin, 2008; Frampton and Lemon, 2009). The reliance on a user simulator for policy optimisation has the major advantage of allowing the learning agent to explore millions of dialogue trajectories on a scale that would be impossible to achieve with real users. Simulated interactions run however the risk of deviating from real user behaviours.

A limitation faced by MDP approaches is the assumption that the dialogue state is fully observable. As frequently noted in the course of this thesis, this assumption simply does not hold for most dialogue domains, owing to the presence of multiple sources of uncertainty, in particular speech recognition errors. An elegant solution to this problem is to extend the MDP framework by allowing the state to be a hidden variable that is indirectly inferred from observations. Such extension gives rise to a *Partially Observable Markov Decision Process* (POMDP). POMDPs are formally defined as tuples $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, Z \rangle$. As in a classical MDP, $\mathcal{S}$ represents the state space, $\mathcal{A}$ the action space, T the transition probability $P(s'|s,a)$ between states, and R the reward function $R(s,a)$. However, the actual state is no longer directly observable. Instead, the process is associated with an observation space $\mathcal{O}$ that expresses the set of possible observations that can be perceived by the system (for instance, the N-best lists of user dialogue acts generated by the speech recogniser and NLU modules) . The function Z finally defines the probability $P(o|s)$ of observing o in the current state s.

In the POMDP setting, the agent knowledge at a given time is represented by the *belief state* b, which is a probability distribution $P(s)$ over possible states. The belief state is continuously updated as additional information becomes available in the form of e.g. new observations. Based on this belief state, a POMDP policy is defined as a function mapping each possible belief state to its optimal action. As for MDP-based reinforcement learning, POMDP approaches usually derive the dialogue policy from interactions with a user simulator (Young et al., 2010; Thomson and Young, 2010; Daubigney et al., 2012b). The optimisation process is however considerably more complex than for MDPs, as the belief state is a continuous and high-dimensional structure. Approximation techniques are therefore necessary in order to extract dialogue policies of reasonable quality in such complex space. The next chapter fleshes out the theoretical foundations of these modelling strategies and their applications to spoken dialogue systems.

Benefits and limitations of statistical approaches

As stated in the previous sections, one key benefit of statistical approaches is the improved robustness towards errors and unexpected events. This robustness stems primarily from the use of probabilistic reasoning techniques that explicitly account for the uncertainty inherent in spoken dialogue. The second benefit is the possibility to optimise dialogue policies in a principled, data-

driven manner based on a generic specification of the system objectives expressed in the reward function. This specification allows the system designer to explicitly encode the various goals and costs of the system. This ability to represent trade-offs between multiple, sometimes conflicting objectives is one important advantage of reinforcement learning approaches. Empirical studies have shown that automatically optimised policies can outperform hand-crafted strategies in both simulated environments and real user trials, based on objectives and subjective metrics of dialogue success (Lemon and Pietquin, 2007; Young et al., 2013).

Statistical modelling techniques come however with a number of challenges of their own. The most pressing issue is the paucity of suitable data sets. Statistical models often require large amounts of training data to estimate their parameters. Unfortunately, real interaction data is scarce, expensive to acquire, and difficult to transfer from one domain to another. User simulators can partly alleviate this problem, but must often themselves be bootstrapped from data, and offer no guarantee of producing conversational behaviours that reflect those of real users. The computational complexity of the learning algorithm can also be problematic. Statistical approaches – and especially POMDP-based systems – must often carefully engineer their state and action variables to limit the size of the search space and ensure the learning process remains tractable. Albeit several dimensionality reduction techniques have been proposed in the literature (Williams and Young, 2005; Young et al., 2010; Cuayáhuitl et al., 2010; Crook and Lemon, 2011), most work has so far concentrated on slot-filling applications. Domains such as tutoring systems, cognitive assistants and human-robot interaction must however often deal with state-action spaces that are considerably more elaborate, with multiple tasks to perform, sophisticated user models, and a complex, dynamic environment. In such settings, the dialogue system might need to track a large number of variables in the course of the interaction, which quickly leads to a combinatorial explosion of the state space. How to define appropriate statistical models for these open-ended dialogue domains remains an open question, to which the present thesis aims to offer preliminary answers.

Finally, many practical dialogue applications need to enforce generic constraints on the dialogue flow. Such constraints may for instance correspond to business rules specific to the particular application. The incorporation of such constraints in the optimisation process of dialogue policies is however far from trivial. As noted by Paek and Pieraccini (2008), this lack of direct control on the final policy is one of the main reasons for the slow adoption of RL approaches in industrial systems. Although some researchers have worked on the integration of expert knowledge into dialogue policy learning (Heeman, 2007; Williams, 2008b), much work remains to be done to bring about a unified approach to dialogue management that combines the robustness of data-driven approaches with the control and expressivity of hand-crafted strategies.

Table 2.1 presents a comparison of the most important hand-crafted and statistical methods to dialogue management in terms of state representation, account of state uncertainty (in the sense of having multiple hypotheses about the current state, each one assigned with a specific probability), type of state update and action selection mechanism. The last row also describes how the approach developed in this thesis stands in comparison to these methods.

2.4 Summary

We have presented in this chapter the most important concepts and methods in the area of dialogue processing and management. Starting with a linguistic analysis of the most important dialogue

Approach	State representation	State uncertainty	State update mechanism	Action selection mechanism
Finite state automata	Atomic state	no	Traversal of matching edge	Action associated with node
Frame-based systems (e.g. Seneff and Polifroni, 2000)	Slot/value pairs	no	Slot-filling given user inputs	Production rules
Information state update (e.g. Larsson and Traum, 2000a)	Rich typed feature structure	no	Update rules	Decision rules
Plan-based systems (e.g. Freedman, 2000; Allen et al., 2001)	BDI model	no	Plan recognition and update of BDI model	Classical planning
Supervised approaches (e.g. Hurtado et al., 2005)	Atomic/factored state	no	Extraction of state variables from history and task status	Classifier estimated from Wizard-of-Oz data by supervised learning
MDP-based systems (e.g. Walker, 2000; Levin et al., 2000)	Atomic/factored state	no	Extraction of state variables from history and task status	Policy optimised via reinforcement learning from real or simulated dialogues
POMDP-based systems (e.g. Roy et al., 2000; Young et al., 2010)	Atomic/factored state	yes	Update of belief state	Policy optimised via reinforcement learning from real or simulated dialogues
Probabilistic rules	Factored state	yes	Structured belief state update (with probabilistic rules)	Policy optimised via (Bayesian) supervised or reinforcement learning

Table 2.1: Comparison of dialogue management approaches.

phenomena, we discussed several key aspects of verbal interactions, such as their articulation in sequences of turns and dialogue acts. We also stressed the importance of contextual knowledge in the interpretation and production of dialogue acts, and the role of grounding signals to maintain mutual understanding among the conversational partners.

Section 2.2 described how spoken dialogue systems are internally structured. As we have explained, dialogue systems are often instantiated in complex software architectures that comprise numerous interconnected components for tasks such as speech recognition, understanding, dialogue management, natural language generation and speech synthesis. Dialogue systems can also be extended to handle (i.e. both perceive and act upon) extra-linguistic modalities and environmental factors. The range of possible applications of dialogue system technology is broad and includes domains as varied as mobile applications for information access and service delivery, in-car navigation systems, smart home environments, cognitive assistants, tutoring systems, and service robots.

The last section presented an overview of the dialogue management task. A key concept shared by virtually all approaches to dialogue management is the *dialogue state*, a data structure used to encode the system knowledge of the current conversational situation. This dialogue state can take multiple forms, from the atomic symbols used in finite-state approaches to the rich nested feature structures employed in information state formalisms. Based on this dialogue state, an action selection mechanism is then responsible for the selection of the next action to execute. In hand-crafted approaches, this mechanism is manually specified by the application developer, either via direct mappings from state to actions, or indirectly through the use of planning techniques. Statistical approaches, on the other hand, seek to automatically optimise dialogue policies from (real or simulated) interaction data. A wide range of learning techniques have been developed to perform this optimisation, from supervised learning on a Wizard-of-Oz data set to reinforcement learning based on a user simulator and a generic reward function. Reinforcement learning techniques can themselves be divided into MDP approaches, where action effects are stochastic but the dialogue state itself is assumed to be known, and POMDP approaches, which incorporate both stochastic action effects and state uncertainty.

We concluded our review of dialogue management approaches by noting that both hand-crafted and statistical methods have significant challenges to address. This is especially striking for open-ended domains such as human-robot interaction, which exhibit both high levels of noise and uncertainty and a rich dialogue context. One of the central claims of this thesis is that these domains are best addressed with a hybrid approach to dialogue management that combines probabilistic modelling with expert knowledge about the domain structure. Chapter 4 presents how such modelling approach can be formalised. But before doing so, we first need to lay down the mathematical apparatus required for designing probabilistic models of dialogue, which is the subject of the next chapter.

Chapter 3

Probabilistic models of dialogue

The previous chapter provided an overview of the most influential approaches to dialogue management, and outlined in particular the benefits of statistical techniques to account for the uncertainty and unpredictability inherent to spoken dialogue. The present chapter exposes the theoretical and methodological foundations of these statistical approaches as well as their application to the dialogue management problem.

The first section of this chapter concentrates on the use of graphical models to design structured representations of probability and utility distributions. Graphical models provide mathematically principled methods for representing, estimating and reasoning over complex probabilistic problems. The section starts with the most well-known type of directed graphical model, namely Bayesian networks. We then show how to extend Bayesian networks to capture temporal sequence and express decision-theoretic problems through actions and utilities. We also review the most important inference and learning algorithms developed for these graphical models.

Based on these graphical models, the second section presents the fundamental principles of reinforcement learning, which is the learning framework employed by most statistical approaches to dialogue management. The section starts with a definition of Markov Decision Processes and explains how policies can be automatically optimised for such processes. The discussion is then extended to partially observable environments in which the current state is hidden and must be indirectly inferred from observations.

Once the mathematical foundations of graphical models and reinforcement learning are in place, the third and last section of this chapter describe how these concepts and techniques can be practically applied to model dialogue management tasks. The section provides a survey of the multiple approaches that have been developed in the last two decades to automatically optimise dialogue policies based on various flavours of supervised and reinforcement learning.

3.1 Graphical models

We describe in this section the core properties of (directed) graphical models,[1] their formal representation, and their use in learning and inference tasks.

[1] There also exists a variety of undirected graphical models, amongst which Markov networks, as well as partially directed models such as Conditional Random Fields, but they will not be discussed nor employed in this thesis.

3.1.1 Representations

Bayesian networks

Let $\mathbf{X} = X_1...X_n$ denote a set of random variables, where each variable X_i is associated with a range of mutually exclusive values. This range can be either discrete or continuous. For dialogue models, the range of a variable X_i is typically discrete and can be explicitly enumerated. The enumeration of values for the variable X_i can be written $Val(X_i) = \{x_i^1, ..., x_i^m\}$.

In the general case, the variables $\mathbf{X}$ can be interrelated by complex probabilistic dependencies. These dependencies can be expressed through the joint probability distribution $P(X_1...X_n)$. The size of this joint distribution is however exponential in the number n of variables, and is therefore difficult to manipulate (let alone estimate and reason over) directly.

It is fortunately possible to exploit conditional independence properties to reduce the complexity of the joint probability distribution. For three disjoint sets of random variables $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$, we say that $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{Z}$ iff $P(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z}) = P(\mathbf{X} \mid \mathbf{Z})P(\mathbf{Y} \mid \mathbf{Z})$ for all possible combination of values for $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$. This conditional independence is denoted $(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$.

Conditional independence allows a joint probability distribution to be decomposed into smaller distributions that are much easier to work with. For a variable X_i in $X_1...X_n$, we can define the set $parents(X_i)$ as the minimal set of predecessors of X_i such that the other predecessors of X_i are conditionally independent of X_i given $parents(X_i)$. The set of parents can be empty if the variable X_i is independent of all other variables. This definition enables us to decompose the joint distribution based on the chain rule:

$$P(X_1, ...X_n) = \prod_{i=1}^{n} P(X_i \mid X_1, ...X_{i-1}) \tag{3.1}$$

$$= \prod_{i=1}^{n} P(X_i \mid parents(X_i)) \tag{3.2}$$

This decomposition can be graphically represented in a *Bayesian network*. A Bayesian network is a directed acyclic graph (DAG) where each random variable is represented by a distinct node. These nodes are connected via directed edges that reflect conditional dependencies. In other words, an edge $X_m \rightarrow X_n$ indicates that $X_m \in parents(X_n)$. Each variable X_i in the Bayesian network must be associated with a specific conditional probability distribution $P(X_i \mid parents(X_i))$. Together with the directed graph, the conditional probability distributions (abbreviated as CPDs) fully determine the joint probability distribution of the Bayesian network.

Given such definitions, the Bayesian network can be directly used for inference by querying the distribution of a subset of variables, often given some additional evidence. Two operations are especially useful when manipulating probability distributions:

- Marginalisation (also called "summing out"), which derives the probability of the variable X given its conditional distribution $P(X \mid Y)$ and the distribution $P(Y)$:

$$P(X) = \sum_{y \in Val(Y)} P(X, Y) = \sum_{y \in Val(Y)} P(X \mid Y)P(Y) \tag{3.3}$$

- Bayes' rule, which reverses the order of a conditional distribution between two variables X and Y (possibly with some background evidence $\mathbf{e}$):

$$P(X \mid Y, \mathbf{e}) = \frac{P(Y \mid X, \mathbf{e}) P(X \mid \mathbf{e})}{P(Y \mid \mathbf{e})} \tag{3.4}$$

As an illustration, Figure 3.1 provides an example of Bayesian network that models the probability of occurrence of a fire at a given time. The probability of this event is dependent on the current weather. In addition, two (imperfect) monitoring systems are used to detect possible fires; one on the ground, and one via satellite.

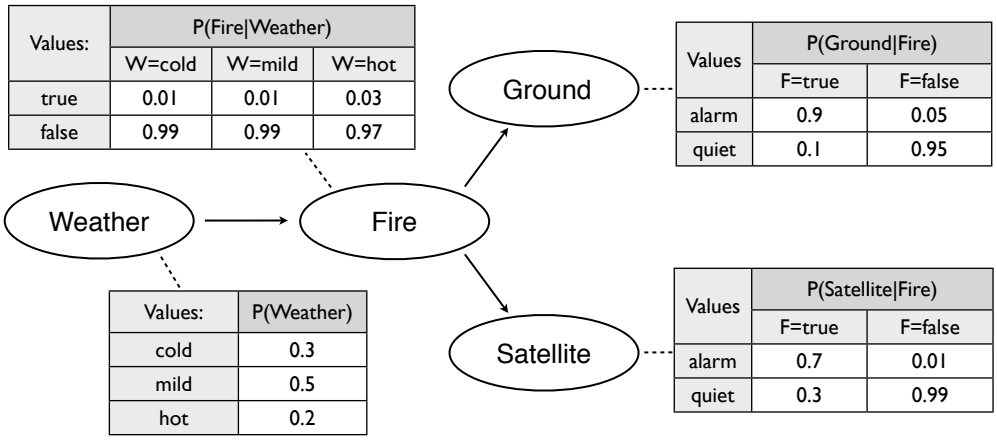


Figure 3.1: Example of Bayesian network with four random variables.

There is one distinct distribution for every combination of values in $parents(X_i)$. The probabilistic model defined in the figure includes therefore a total of eight distributions. The distributions in Figure 3.1 are *categorical* distributions.[2] Categorical distributions can be encoded with look-up tables that map every possible value in $Val(X_i)$ to a particular probability. Many of the probability distributions used throughout this thesis will take the form of categorical distributions. Various other representations for discrete CPDs are however conceivable, such as deterministic distributions and distributions based on independence of causal influence (Díez and Druzdzel, 2006).

A Bayesian network can also contain continuous distributions. These distributions are usually encoded with *density functions* represented in a parametric form. A well-known example of parametric distribution is the normal distribution $\mathcal{N}(\mu, \sigma^2)$, which is defined by its two parameters μ and σ^2. Continuous distributions can also expressed with non-parametric methods such as Kernel Density Estimation (KDE). Finally, hybrid models involving both discrete and continuous variables can be defined. The reader is invited to refer to Bishop (2006) and Koller and Friedman (2009) for more details on parametric and non-parametric distributions. Appendix A enumerates the most important discrete and continuous probability distributions used in this thesis.

[2]Categorical distributions are often conflated with *multinomial* distributions, which specify the number of times an exclusive event will occur in a repeated independent trial with k possible categories, with each category having a given fixed probability. A categorical distribution is equivalent to a multinomial distribution for a single observation.

Reasoning over time

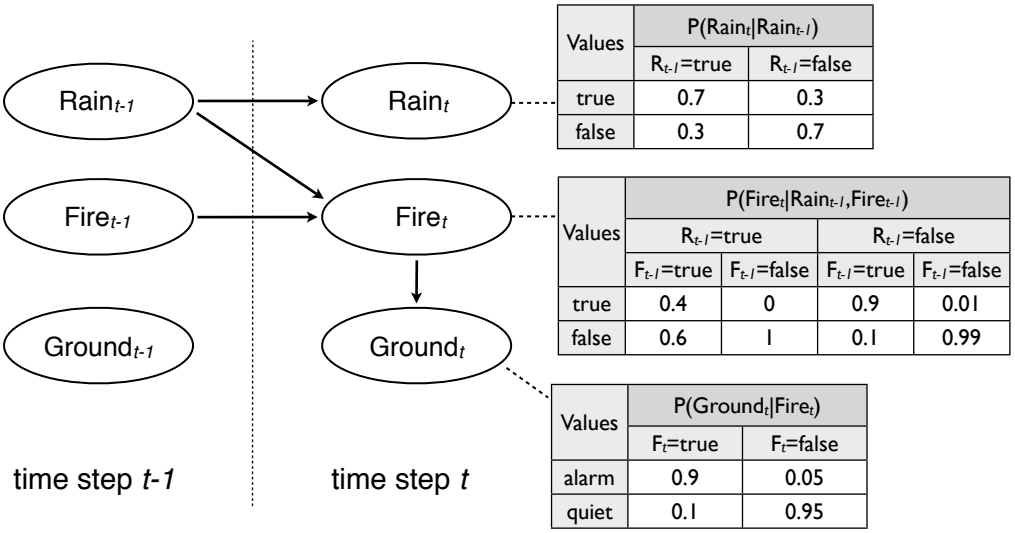
In order to apply Bayesian networks to tasks such as dialogue management, two additional elements are necessary. The first extension is to allow variables to evolve as a function of time. Such temporal dependencies are indeed necessary to account for the dynamic nature of dialogue (the dialogue state is not a static entity but is expected to change over time). Two assumptions are usually made to structure such temporal dependencies:

1. The first assumption, called the Markov assumption, is that the variable values at time t only depend on the previous time slice $t-1$. Formally, let $\mathbf{X}$ be an arbitrary collection of variables. We denote by $\mathbf{X}_t$ the random variables that express their values at time t. The Markov assumption states that $(\mathbf{X}_t \perp \mathbf{X}_{0:(t-2)} \mid \mathbf{X}_{t-1})$.

2. The second assumption is that the process is stationary[3] – that is, that the probability $P(\mathbf{X}_t \mid \mathbf{X}_{t-1})$ is the same for all values of t.

Given these two assumptions, we can define a stochastic process with a probability distribution $P(\mathbf{X}_t \mid \mathbf{X}_{t-1})$ that specifies the distribution of the variables $\mathbf{X}$ at time t given their values at time $t-1$. Such model is called a *dynamic Bayesian network* (DBN). The distribution $P(\mathbf{X}_t \mid \mathbf{X}_{t-1})$ can be internally factored and include dependencies both between the time slices $t-1$ and t and within the slice t. Figure 3.2 shows a concrete example of dynamic Bayesian network. The DBN provides a factored representation of the distribution $P(R_t, F_t, G_t \mid R_{t-1}, F_{t-1})$.

Given the specification of the distribution $P(\mathbf{X}_t \mid \mathbf{X}_{t-1})$ and an initial distribution $P(\mathbf{X}_0)$, a dynamic Bayesian network can be "unrolled" onto multiple time slices. This unrolled model corresponds to a classical Bayesian network.



Values	P(Rain$_t$|Rain$_{t-1}$)	
	R$_{t-1}$=true	R$_{t-1}$=false
true	0.7	0.3
false	0.3	0.7

Values	P(Fire$_t$|Rain$_{t-1}$,Fire$_{t-1}$)			
	R$_{t-1}$=true		R$_{t-1}$=false	
	F$_{t-1}$=true	F$_{t-1}$=false	F$_{t-1}$=true	F$_{t-1}$=false
true	0.4	0	0.9	0.01
false	0.6	1	0.1	0.99

Values	P(Ground$_t$|Fire$_t$)	
	F$_t$=true	F$_t$=false
alarm	0.9	0.05
quiet	0.1	0.95

Figure 3.2: Example of dynamic Bayesian network.

[3] A *stationary* process must be distinguished from a *static* process: a static process is a stochastic process that remains constant for all time steps. In contrast, a stationary process can change over time, but the transition model that describes the dynamics of this process remains constant.

Decision problems

Dynamic Bayesian networks are well-suited to represent temporal processes. However, in sequential decision tasks such as dialogue management, tracking the current state is only the first step of the reasoning process. The agent must also be able to calculate the relative utilities of the various actions that can be executed at that state. The second extension of Bayesian networks thus pertains to the inclusion of actions and utilities in addition to state variables.

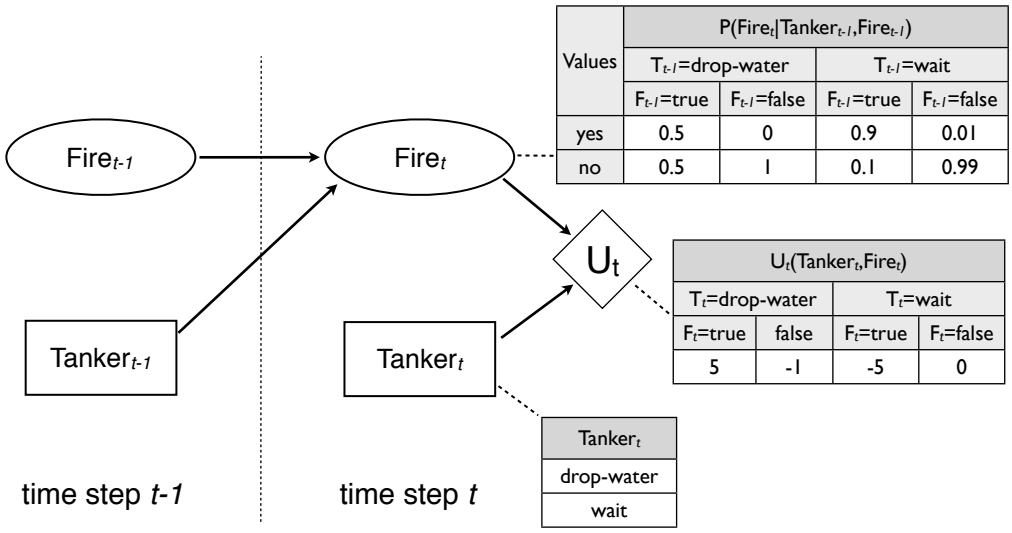
Decision networks[4] are Bayesian networks augmented with a representation of action variables and their corresponding utilities. Decision networks may include three classes of nodes:

1. *Chance nodes* correspond to the classical random variables described so far. Chance nodes are associated with CPDs that define the relative probabilities of the node values given the values in the parent nodes.

2. *Decision nodes* (sometimes also called action nodes) correspond to variables that are under the control of the system. The values of these nodes reflect an active choice made by the system to execute particular actions.

3. *Utility nodes* express the utilities (from the system's point of view) associated with particular situations expressed in the node parents. Typically, these parents combine both chance and decision variables. Utility nodes are coupled with utility distributions that associate each combination of values in the node parents with a specific (negative or positive) utility.

Decision networks combined with temporal dependencies are called *dynamic decision networks*. Figure 3.3 illustrates a dynamic decision network with a decision variable containing two alternative actions assigned to distinct utility values depending on the occurrence of fire.



Values	P(Fire$_t$|Tanker$_{t-1}$,Fire$_{t-1}$)			
	T_{t-1}=drop-water		T_{t-1}=wait	
	F_{t-1}=true	F_{t-1}=false	F_{t-1}=true	F_{t-1}=false
yes	0.5	0	0.9	0.01
no	0.5	1	0.1	0.99

U_t(Tanker$_t$,Fire$_t$)			
T_t=drop-water		T_t=wait	
F_t=true	false	F_t=true	F_t=false
5	-1	-5	0

Figure 3.3: Example of dynamic decision network. By convention, chance nodes are represented with circles, decision nodes with squares, and utility nodes with diamonds.

[4]Decision networks are also called *influence diagrams*.

3.1.2 Inference

Generalities

The main purpose of probabilistic graphical models is to evaluate *queries* – that is, calculate a posterior distribution over a subset of variables, usually given some evidence. Assuming a graphical model defining the joint probability distribution of a set of variables $\mathbf{X}$, a probability query is a posterior distribution of the form $P(\mathbf{Q} \mid \mathbf{E}=\mathbf{e})$, where $\mathbf{Q} \subset \mathbf{X}$ denotes the query variables, $\mathbf{E} \subset \mathbf{X}$ the evidence variables, and $\mathbf{e}$ a possible assignment of values for the evidence variables. If the set of evidence variables is empty, the query is reduced to the calculation of a marginal distribution. Graphical models augmented with decision and utility variables can also be used to answer utility queries of the form $U(\mathbf{Q} \mid \mathbf{E} = \mathbf{e})$. In this case, the query variables often correspond to decision nodes whose utility is to be estimated.

A wide range of inference algorithms have been developed to efficiently evaluate these probability and utility queries. These algorithms can be either exact or approximate.

Exact algorithms calculate the precise posterior distribution corresponding to the query through a sequence of manipulation operations on the CPDs contained in the graphical model. One popular algorithm for exact inference is variable elimination (Zhang and Poole, 1996). Variable elimination relies on dynamic programming techniques to evaluate a query through a a sequence of matrix operations (summation and pointwise product). These operations are defined on so-called "factors" that represent CPDs in a matrix format. Variable elimination can be generalised to handle utility queries using joint factors (Koller and Friedman, 2009). Other algorithms such as message passing on clique trees can also be used (Jensen et al., 1990).

Exact inference remains unfortunately difficult to scale to large, densely interconnected graphical models, and approximate techniques are often unavoidable in many practical domains. Algorithms for approximate inference in graphical models include approaches such as loopy belief propagation (Murphy et al., 1999), variational methods (Jordan et al., 1999), and a wide array of sampling techniques (MacKay, 1998), sometimes also called Monte Carlo methods. Popular sampling techniques include various flavours of importance sampling (Fung and Chang, 1989; Cheng and Druzdzel, 2000) and Markov Chain Monte Carlo (MCMC) approaches such as Gibbs sampling (Pearl, 1987; Gamerman and Lopes, 2006).

Probabilistic inference is a difficult computational task. In fact, inference on unconstrained Bayesian Networks is known to be #P-hard, which is a complexity class that is strictly harder than NP-complete problems. This holds both for exact inference (Cooper, 1990), and – perhaps more surprisingly – also for approximate inference (Dagum and Luby, 1993).

The `openDial` toolkit we have developed for this thesis includes two inference algorithms: generalised variable elimination (Koller and Friedman, 2009, p. 1103) and a specific type of importance sampling algorithm called likelihood weighting, which we outline below. These algorithms are used in the processing workflow of the dialogue manager to (1) update the dialogue state upon the reception of new observations and (2) select system actions on the basis of this updated dialogue state (the details of this workflow is provided in Section 4.4).

Likelihood weighting

To make our discussion of inference frameworks for graphical models more concrete, we describe below a simple but efficient sampling method called *likelihood weighting* (Fung and Chang, 1989),

which we have used as inference algorithm for most of the experiments conducted in this thesis.

The intuition behind sampling algorithms is to estimate the posterior distribution expressed in the query by collecting a large quantity of samples (i.e. assignments of values to the random variables) drawn from the graphical model. Likelihood weighting proceeds by sampling the random variables in the graphical model one by one, in topological order (i.e. from parents to children).[5] For instance, sampling the network in Figure 3.1 will start with the variable $Weather$, then $Fire$ (based on the value drawn for the parent $Weather$), and finally $Ground$ and $Satellite$ (based on the value drawn for $Fire$). In order to account for the evidence $\mathbf{e}$, every sample is associated with a specific *weight* that expresses the likelihood of the evidence given the assignment for all the other variables. For graphical models that include utility variables, samples also record the total utility accumulated for the sampled values. The pseudocode in Algorithm 3.1.2 outlines the sampling procedure.

Algorithm 1 : GET-SAMPLE $(\mathcal{B}, \mathbf{E} = \mathbf{e})$

Input: Bayesian/decision network $\mathcal{B}$ over $\mathbf{X}$ with topological ordering $X_1, ...X_n$
Input: Evidence $\mathbf{E} = \mathbf{e}$
Output: Full sample drawn from $\mathcal{B}$ together with a weight and utility

```
Initialise sample x ← ⟨e⟩
Initialise weight w ← 1 and utility u ← 0
for all X_i ∈ X_1, ...X_n do
  if X_i is a chance variable and X_i ∈ E then
    w ← w × P(X_i = e(X_i) | x)
  else if X_i is a chance or decision variable and X_i ∉ E then
    x_i ← random sample value drawn from P(X_i | x)
    x ← x ∪ ⟨x_i⟩
  else if X_i is a utility variable then
    u ← u + U_i(x)
  end if
end for
return x, w, u
```

The notation $\mathbf{e}(X_i)$ refers to the value specified for the variable X_i in the assignment $\mathbf{e}$.

A large number of samples can be collected in such manner. Once enough samples are collected (or the inference engine has run out of time) the resulting posterior distribution for the query variables $\mathbf{Q}$ is derived by normalising the weighted counts associated with each value of the query variables, as shown in Algorithm 3.1.2. Algorithm 3.1.2 extends the procedure to the calculation of utility distributions. In this case, the utility values are not normalised but correspond to a weighted average of the sampled utilities.

3.1.3 Learning

We have so far pushed aside the question of how the distributions in the graphical model are exactly estimated. Early approaches often relied on distributions elicited from human experts based on plausible associations they have observed. Although useful in domains where no data is available,

[5] A partial order on the nodes in the graph can always be found since the network is a directed acyclic graph.

Algorithm 2 :LIKELIHOOD-WEIGHTING $(\mathcal{B}, \mathbf{Q}, \mathbf{E} = \mathbf{e}, N)$

Input: Bayesian/decision network $\mathcal{B}$ over $\mathbf{X}$ with topological ordering $X_1, ...X_n$
Input: Set of query variables $\mathbf{Q}$ and evidence $\mathbf{E} = \mathbf{e}$
Input: Number N of samples to draw
Output: Approximate posterior distribution $P(\mathbf{Q} \mid \mathbf{E} = \mathbf{e})$ given $\mathcal{B}$

Let $\mathbf{W}$ be vectors of weighted counts for each possible value of $\mathbf{Q}$, initialised to zero
for $i = 1 \rightarrow N$ **do**
  $\mathbf{x}, w \leftarrow$ GET-SAMPLE$(\mathcal{B}, \mathbf{e})$
  $\mathbf{W}[\mathbf{x}(\mathbf{Q})] \leftarrow \mathbf{W}[\mathbf{x}(\mathbf{Q})] + w$
end for
Normalise the weighted counts in $\mathbf{W}$
return $\mathbf{W}$

Algorithm 3 : LIKELIHOOD-WEIGHTING-UTILITY $(\mathcal{B}, \mathbf{Q}, \mathbf{E} = \mathbf{e}, N)$

Input: (see above)
Output: Approximate utility distribution $U(\mathbf{Q}, \mathbf{E} = \mathbf{e})$ given $\mathcal{B}$

Let $\mathbf{W}, \mathbf{U}$ be vectors of weighted counts for each possible value of $\mathbf{Q}$, initialised to zero
for $i = 1 \rightarrow N$ **do**
  $\mathbf{x}, w, u \leftarrow$ GET-SAMPLE$(\mathcal{B}, \mathbf{e})$
  $\mathbf{W}[\mathbf{x}(\mathbf{Q})] \leftarrow \mathbf{W}[\mathbf{x}(\mathbf{Q})] + w$
  $\mathbf{U}[\mathbf{x}(\mathbf{Q})] \leftarrow \mathbf{U}[\mathbf{x}(\mathbf{Q})] + w \times u$
end for
Average the weighted utility counts $\mathbf{U}(\mathbf{q}) \leftarrow \frac{\mathbf{U}(\mathbf{q})}{\mathbf{W}(\mathbf{q})}$ $\forall$ values $\mathbf{q}$ of $\mathbf{Q}$
return $\mathbf{U}$

hand-crafted models are unfortunately difficult to scale (only models with a limited number of parameters can be elicited in such manner), and are vulnerable to human errors and inaccuracies. It is therefore often preferable to automatically estimate these distributions from real world data – in other words, via statistical estimation based on a collection of examples in a training set.

Two distinct types of learning tasks can be distinguished:

1. The most common task is *parameter estimation*. Parameter estimation assumes the general structure of the graphical model (i.e. the dependencies between variables) is known, but not the parameters of the individual CPDs. Most discrete and continuous distributions are indeed "parametrised" – that is, they depend on the specification of particular parameters that define the exact shape of the distribution. A categorical distribution on k values has for instance k parameters that assign the relative probability of each outcome. Similarly, a normal distribution $\mathcal{N}(\mu, \sigma^2)$ is governed by its two parameters μ and σ^2.

2. The second possible learning task is *structure learning*. In structure learning, the agent must learn both the structure (i.e. the directed edges) and the parameters of the graphical model, given only the list of variables and the training data. This task is significantly more complex than parameter estimation, as the agent must simultaneously find which variables influence one another and estimate their corresponding conditional dependencies.

We shall concentrate in this thesis on the parameter estimation problem, which is by far the most common type of learning problem in dialogue management.[6]

Maximum likelihood estimation

The most straightforward parameter estimation method is maximum likelihood estimation (MLE). Maximum likelihood estimation searches for the parameters values that provide the best "fit" for the provided data set. In other words, the parameters are set to the values that maximise the likelihood of the observed data. Given a data set $\mathcal{D}$, a graphical model and a set of parameters $\boldsymbol{\theta}$ to estimate in this model, the MLE learning objective is to find the values $\boldsymbol{\theta}^*$ that maximise the probability $P(\mathcal{D}\,;\,\boldsymbol{\theta})$, which is the likelihood of the data set $\mathcal{D}$ given the specified parameter values for $\boldsymbol{\theta}$. This likelihood is often written in logarithmic form:

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} P(\mathcal{D}\,;\,\boldsymbol{\theta}) = \operatorname*{argmax}_{\boldsymbol{\theta}} \log P(\mathcal{D}\,;\,\boldsymbol{\theta}) \tag{3.5}$$

If the data samples cover the complete set of variables in the model, this likelihood can be neatly decomposed in a set of local likelihoods, one for each CPD, and the $\boldsymbol{\theta}^*$ values can be derived in closed-form. For a categorical distribution, the MLE estimates will simply correspond to the relative counts of occurrences in the training data.

The learning problem becomes more complex for partially observed domains in which the data samples contain hidden variables. For the Bayesian network in Figure 3.1, an example of partially observed sample is $\langle Weather = mild, Ground = alarm, Satellite = quiet \rangle$, where the occurrence of fire is not specified. In such cases, the likelihood function is no longer decomposable, which implies that the maximum likelihood estimate is not amenable to a closed-form solution. The only alternative is to resort to iterative optimisation methods such as gradient ascent (Binder et al., 1997) and Expectation Maximisation (Green, 1990).

The main drawback of maximum likelihood estimation is its vulnerability to overfitting when learning from small data sets. For instance, if we only had collected one single data point $Weather = cold$ for the previous example, the MLE estimate for the distribution of $P(\,Weather)$ would be $\langle 1, 0, 0 \rangle$. In other words, maximum likelihood estimation does not take into account any prior knowledge about the relative probability of particular parameter hypotheses, which often lead to unreasonable estimates for low frequency events.

Bayesian learning

An alternative to maximum likelihood estimation is *Bayesian learning*. The key idea of Bayesian approaches to parameter estimation is to view the CPD parameters as random variables and to derive their posterior distributions after observing the data. Bayesian learning starts with an initial prior over the range of parameter values and gradually refines this distribution through probabilistic inference based on the observation of the samples in the training data. Each distribution $P(X_i \,|\, parents(X_i))$ with unknown parameters is therefore associated with a parent node $\theta_{X_i|parents(X_i)}$ that define its parameter distribution.

[6] As shall be argued in the next chapters, the graph structure of probabilistic models of dialogue can often be provided by the system designer.

Parameter distributions are typically continuous (since probabilities are continuous values), and often multivariate. Intuitively, we can think of the variable $\theta_{X_i|parents(X_i)}$ as expressing a "distribution over possible distributions".

Based on this formalisation, parameter estimation can be elegantly reduced to a problem of probabilistic inference over the parameters. Given a prior $P(\boldsymbol{\theta})$ on the parameter values and a data set $\mathcal{D}$, the posterior distribution $P(\boldsymbol{\theta} \mid \mathcal{D})$ is given by Bayes' rule:

$$P(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{P(\mathcal{D} \,;\, \boldsymbol{\theta})\, P(\boldsymbol{\theta})}{P(\mathcal{D})} \tag{3.6}$$

Based on Equation 3.6, the posterior distribution $P(\boldsymbol{\theta} \mid \mathcal{D})$ can be calculated with standard inference algorithms for graphical models.

It is often convenient to encode the distributions of the parameter variables as *conjugate priors* of their associated CPD distribution. In such case, the prior $P(\boldsymbol{\theta})$ and posterior $P(\boldsymbol{\theta} \mid \mathcal{D})$ after observing the data points $\mathcal{D}$ are ensured to remain within the same distribution family. In particular, if the distribution of interest is a categorical distribution, its parameter distribution can be encoded with a Dirichlet distribution, which is known as the conjugate prior of categorical and multinomial distributions. A Dirichlet distribution is a continuous, multivariate distribution of dimension k (with k being the size of the multinomial) that is itself parametrised with so-called concentration hyperparameters denoted $\boldsymbol{\alpha} = [\alpha_1, ...\alpha_k]$. Additional details about the formal properties of Dirichlet distributions can be found in Appendix A.

Figure 3.4 illustrates this Bayesian approach to parameter estimation for the variable $Weather$. As the variable possesses three alternative values, the allowed values for the parameter $\boldsymbol{\theta}_{Weather}$ are three-dimensional vectors $[\theta_{\text{Weather}(1)}, \theta_{\text{Weather}(2)}, \theta_{\text{Weather}(3)}]$, with the standard constraints on probability values: $\theta_{\text{Weather}(i)} \geq 0$ for $i = \{1, 2, 3\}$ and $\theta_{\text{Weather}(1)} + \theta_{\text{Weather}(2)} + \theta_{\text{Weather}(3)} = 1$. As we can observe in the figure, these constraints effectively limit the range of possible values to a 2-dimensional simplex. The $\boldsymbol{\alpha}$ hyperparameters can be intuitively interpreted as "virtual counts" of the number of observations in each category. In Figure 3.4, we can see that the hyperparameters $[5, 10, 5]$ lead to higher probability densities for parameters around the peak $\langle 0.25, 0.5, 0.25 \rangle$. As the number of observations increases, the Dirichlet distribution will gradually concentrate on a particular region of the parameter space until convergence.

In the case of completely observed data, Bayesian learning over several parameters can be decomposed into independent estimation problems (one for each parameter variable):

$$P(\mathcal{D} \,;\, \boldsymbol{\theta}) = \prod_{\theta_i \in \boldsymbol{\theta}} P(\mathcal{D} \,;\, \theta_i) \tag{3.7}$$

As in the maximum likelihood estimation case, the learning task becomes more complicated when dealing with partially observed data, as the posterior distribution can no longer be represented as a product of independent posteriors over each parameter. In this setting, the full posterior is often too complex to be amenable to an analytic solution. Sampling techniques can however be applied to offer reasonable approximations of this posterior. As we shall see in Chapter 5 and 6, the work presented in this thesis is directly grounded in such approximate Bayesian learning methods.

Table 3.1 briefly summarises the parameter estimation methods discussed in this section.

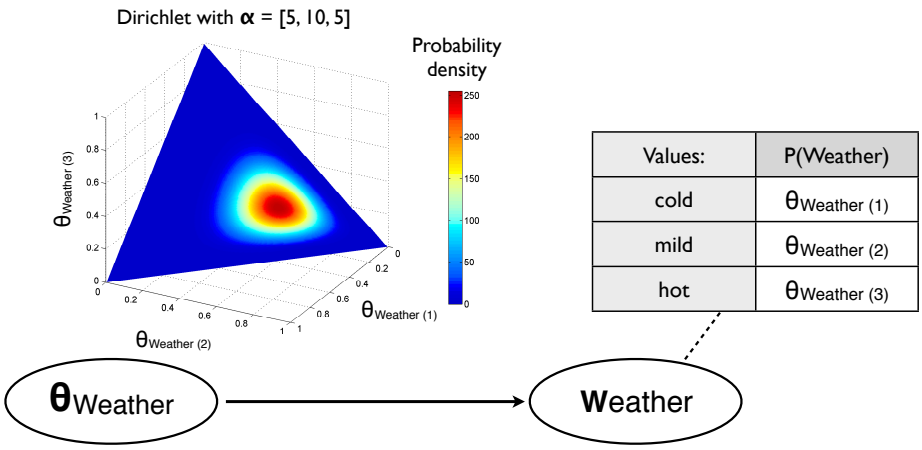


Figure 3.4: Bayesian network with variable $Weather$ and associated parameter $\theta_{Weather}$. As $Weather$ is a categorical distribution, the distribution $P(\theta_{Weather})$ is expressed as a Dirichlet with three dimensions that reflect the relative probabilities for the three values in $Val(Weather)$.

Training data	**Maximum likelihood estimation**	**Bayesian learning**
Fully observed	Maximisation of local likelihood functions	Query on local posterior distribution over each parameter
Partially observed (hidden variables)	Iterative optimisation of global likelihood function	Query on full posterior over parameters via sampling

Table 3.1: Summary of parameter estimation approaches for directed graphical models.

3.2 Reinforcement learning

Dialogue management is fundamentally a problem of sequential decision-making under uncertainty. The objective of the dialogue system is to select the action that is expected to be "optimal" – that is, that yields the maximum long-term utility for the agent. However, in many domains (including dialogue domains), the agent has no knowledge of the internal dynamics of the environment it finds itself in. It must therefore determine the best action to execute in any given state via a process of trial and error. Such learning process is called *reinforcement learning* (RL). It is worth noting that reinforcement learning effectively strikes a middle ground between supervised and unsupervised learning. Contrary to supervised learning, the framework does not require the provision of "gold standard" examples. However, thanks to the reward function it receives from the environment, the agent is able to get a sense of the quality of its own decisions, an element which is absent in unsupervised learning methods.

We provide here a brief introduction to the central concepts in reinforcement learning, and refer the interested reader to Sutton and Barto (1998) for more details.

3.2.1 Markov Decision Processes

Reinforcement learning tasks are typically based on the definition of a *Markov Decision Process* (MDP), which is a tuple $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$ where:

- $\mathcal{S}$ is the state space of the domain and represents the set of all (mutually exclusive) states.
- $\mathcal{A}$ is the action space and represents the possible actions that can be executed by the agent.
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition function and encodes the probability $P(s'|s, a)$ of reaching state s' after executing action a in state s.
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \Re$ is the reward value associated with the execution of action a in state s.

Markov Decision Processes can be explicitly represented as dynamic decision networks. As we can see from the graphical illustration in Figure 3.5, the state at time $t+1$ is dependent both on the previous state at time t and the action a_t performed by the system. After each action, the system received a reward $r_t = R(s_t, a_t)$ that depends both on the state and selected action.

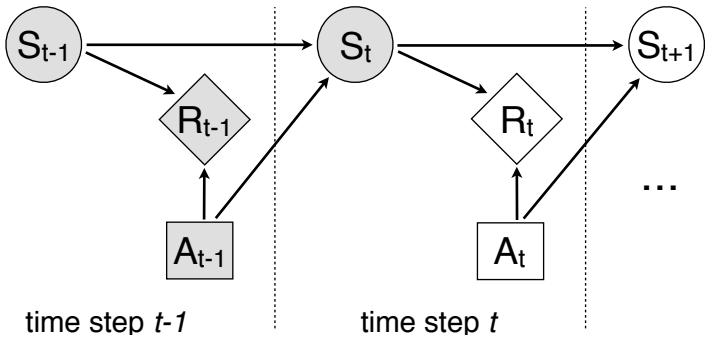


Figure 3.5: Graphical model of a Markov Decision Process (MDP) unfolded on a few time steps. Greyed entities indicate observed variables. In the MDP case, all past states, actions and rewards are observed, as well as the current state.

Given a particular MDP problem, the goal of the learning agent is to find an optimal policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ that maps each possible state to the best action to execute at that state. The best action is the action that maximises the *expected return* for that state, which is the discounted sum of rewards, starting from the current state up to a potentially infinite horizon. In this sum, a geometric discount factor γ (with $0 \leq \gamma \leq 1$) indicates the relative worth of future rewards in regard to present ones. At one extreme, $\gamma = 0$ corresponds to a short-sighted agent only interested in its immediate reward, while $\gamma = 1$ corresponds to an agent for which immediate and distant rewards are valued equally. For a given policy π, the expected return for an arbitrary state s in $\mathcal{S}$ is expressed through the value function $V^\pi(s)$:

$$V^\pi(s) = E\left\{ r_0 + \gamma r_1 + \gamma^2 r_2 + ... \;\middle|\; s_0 = s, \pi \text{ is followed} \right\} \tag{3.8}$$

$$= E\left\{ \sum_{i=0}^{\infty} \gamma^i r_i \;\middle|\; s_0 = s, \pi \text{ is followed} \right\} \tag{3.9}$$

where $r_i = R(s_i, \pi(s_i))$ is the reward received at time i after performing the action specified by the policy π in state s_i. Equation (3.9) can be rewritten in a recursive form, leading to the well-known Bellman equation (Bellman, 1957):

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) V^\pi(s') \tag{3.10}$$

In other words, the expected return in state s equals its immediate reward plus the expected return of its successor state. The recursive definition offered by the Bellman equation is crucial for many reinforcement learning methods, since it allows the value function to be estimated by an iterative process in which the value function is gradually refined until convergence.

Another useful concept is the action-value function $Q^\pi(s, a)$ that expresses the return expected after performing action a in state s and following the policy π afterwards:

$$Q^\pi(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V^\pi(s') \tag{3.11}$$

The value and action-value functions are closely related, as $V^\pi(s) = Q^\pi(s, \pi(s))$.

The objective of the learning process is to find a policy $\pi^* : \mathcal{S} \to \mathcal{A}$ which selects the action with highest expected return for all possible states: $V^{\pi^*}(s) \geq V^\pi(s)$ for any state s and policy π. For a given MDP, there is at least one policy that satisfies this constraint. The value and action-value functions for this optimal policy are respectively denoted V^* and Q^*.

Two families of reinforcement learning approaches can be used to determine this policy:

1. *Model-based* approaches first estimate an explicit model of the MDP and subsequently optimise a policy based on this model. The agent initially collects data by interacting with its environment and recording the reward and successor state following each action. Based on this data, standard parameter estimation methods (as outlined in Section 3.1.3) can be used to fix the MDP parameters, and the resulting model is applied to extract the corresponding optimal policy via dynamic programming (Bertsekas and Tsitsiklis, 1996) or more advanced Bayesian methods (Dearden et al., 1999). The most well-known model-based learning algorithm is *value iteration*, which operates by estimating the value function via a sequence of updates based on Bellman's equation. Given a value function estimate V_k available at step k, the estimate at step $k + 1$ is calculated as:

$$V_{k+1}(s) = \max_a R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, a)\, V_k(s') \tag{3.12}$$

   Once the iterations have converged, the optimal policy is straightforwardly derived as:

$$\pi^*(s) = \operatorname*{argmax}_a Q^*(s, a) \tag{3.13}$$

2. *Model-free* approaches skip the estimation of the underlying MDP model in order to directly learn Q^* functions from the agent's interaction experience. The most popular model-free techniques are Q-learning (Watkins and Dayan, 1992), SARSA (Rummery, 1995) and gradient descent (Sutton et al., 2009). The main idea behind model-free methods is to let the

agent try out different actions, observe the effects in terms of rewards and successor state, and use this information to refine the estimate of the Q^* function. This operation is repeated for a large number of episodes until convergence.

One key question that must be addressed in both model-based and model-free approaches is how to efficiently explore the space of possible actions. The agent should indeed favour the selection of high-utility actions in most cases (since they are the ones of interest to the agent), but should also occasionally explore actions that are currently thought to be less effective to avoid being stuck in a suboptimal behaviour. This trade-off, called the *exploration-exploitation* dilemma, is one of the central research questions in reinforcement learning.

3.2.2 Partially Observable Markov Decision Processes

A limitation faced by MDP approaches is the assumption that the dialogue state is fully observable. As we have frequently noted in this thesis, this assumption does not hold for most dialogue systems, owing to the presence of multiple sources of uncertainty, due in particular to speech recognition errors. An elegant solution to this problem is to extend the MDP framework by allowing the state to be a hidden variable that is indirectly inferred from observations. Such extension gives rise to a *Partially Observable Markov Decision Process* (POMDP). POMDPs are formally defined as tuples $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, Z \rangle$. As in a classical MDP, $\mathcal{S}$ represents the state space, $\mathcal{A}$ the action space, T the transition probability $P(s'|s, a)$ between states, and R the reward function $R(s, a)$. However, the actual state is no longer directly observable. Instead, the process is associated with an observation space $\mathcal{O}$ that expresses the set of possible observations that can be perceived by the system. The function Z then defines the probability $P(o|s)$ of observing o in the current state s. Figure 3.6 provides a graphical illustration of the POMDP framework. As we can see, POMDPs can also be expressed as dynamic decision networks in which the state variable is not directly observed.

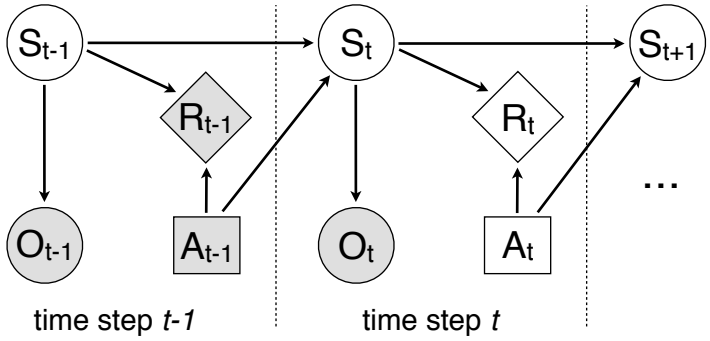


Figure 3.6: Graphical model of a Partially Observable Markov Decision Process (POMDP) unfolded on a few time steps. Compared to Figure 3.5, we notice that the state is no longer directly accessible but must be inferred from the observations and (predicted) state history.

The agent knowledge at a given time is represented by the *belief state* b, which is a probability distribution $P(s)$ over possible states. After each system action a and subsequent observation o, the belief state b is updated to b' in order to incorporate the new information. This belief update is

a simple application of Bayes' theorem:

$$b'(s) = P(s'|b,a,o) = \frac{P(o|s')\ P(s'|b,a)}{P(o|b,a)} \tag{3.14}$$

$$= \eta\ P(o|s') \sum_{s} P(s'|s,a)\ b(s) \tag{3.15}$$

where $\eta = P(o|b,a)$ serves as a normalisation constant and is usually never calculated explicitly.

In the POMDP setting, a policy is a function $\pi = \mathcal{B} \rightarrow \mathcal{A}$ mapping each possible belief state to its optimal action. Mathematically, the belief state space $\mathcal{B}$ is a $(|\mathcal{S}|-1)$-dimensional simplex (where $|\mathcal{S}|$ is the size of the state space), which is a continuous and high-dimensional space. The optimisation of the dialogue policy is therefore considerably more complex than for MDPs. The value function V^{π} for a policy π is the fixed point of Bellman's equation:

$$V^{\pi}(b) = \sum_{s \in S} R(s,a)b(s) + \gamma \sum_{o \in \mathcal{O}} P(o|b,\pi(b))\ V^{\pi}(b') \tag{3.16}$$

where b' is the updated belief state following the execution of action $\pi(b)$ and the observation of o, as in Equation (3.15). The optimal value function $V^*(b)$ for finite-horizon problems is known to be piecewise linear and convex in belief space, as proved by Sondik (1971). The value function can therefore be represented by a finite set of vectors, called α-vectors. Each vector α_i is associated with a specific action $a(i) \in \mathcal{A}$.[7] The vectors are of size $|\mathcal{S}|$ and $\alpha_i(s)$ is a scalar value representing the value of action $a(i)$ in state s. Given these vectors, the value function simplifies to:

$$V^*(b) = \max_i \alpha_i \cdot b \tag{3.17}$$

And the policy π^* can be rewritten as:

$$\pi^*(b) = a\left(\operatorname*{argmax}_i(\alpha_i \cdot b)\right) \tag{3.18}$$

Extracting the α-vectors associated with a POMDP problem is however computationally challenging given the high-dimensional and continuous nature of the belief space, and exact solutions are intractable beyond toy domains. As shown by Papadimitriou and Tsitsiklis (1987), deriving the α-vectors for a given POMDP (a.k.a. "solving" the POMDP) is a PSPACE-complete problem, which means that the best known algorithms will take time $2^{poly(n,h)}$ to solve a problem with n states and a planning horizon h.

A number of approximate solutions have however been developed. One simple strategy is to rewrite the $Q(b,a)$ function in terms of the Q-values for the underlying MDP, as described by Littman et al. (1998):

$$Q(b,a) = \sum_{s \in \mathcal{S}} Q_{MDP}(s,a)b(s) \tag{3.19}$$

Although this approximation can work well in some settings, it essentially rests on the assumption that the state uncertainty will disappear after one action. It is therefore a poor model for

[7] Note that the reverse is not true: each action can be associated with an arbitrary number of vectors.

information-gathering actions – that is, actions that do not change the actual state but might help in reducing state uncertainty.[8]

Many approximations methods focus on simplifying the belief state space, notably through the use of grid-based approximations (Zhou and Hansen, 2001). The idea is to estimate the value function only at particular points within the belief simplex. At runtime, the action-value function for the current belief state b is then approximated to the value for the closest point in the grid according to some distance measure. Instead of using a fixed grid, most recent POMDP solution methods rely on sampling methods to perform local value updates on specific belief points (Pineau et al., 2003; Kurniawati et al., 2008). Belief compression methods have also proved to be useful (Roy et al., 2005). Other optimisation methods directly search in the space of possible policies constrained to a particular form such as finite-state controllers (Hansen, 1998).

A final alternative, which we follow in this thesis, is to rely on online planning algorithms (Ross et al., 2008; Silver and Veness, 2010a). The idea is to let the agent estimate the $Q(b, a)$ values at execution time via look-ahead search on a limited planning horizon. Compared to offline policies, the major advantage of online approaches is that the agent only needs to consider the current state to plan instead of enumerating all possible ones. It can also more easily adapt to changes in the reward or transition models, as the policy is not compiled in advance but generated at runtime. Online planning can moreover be used to simultaneously learn or refine these models during the interaction, as demonstrated by Ross et al. (2011). The available planning time is however more limited, since planning is in this case interleaved with system execution and must therefore meet real-time constraints.

Despite these recent advances, the optimisation of POMDP policies remains nevertheless to a large extent an open research question in the fields of reinforcement learning and decision-theoretic planning. One important insight that transpires in much of the POMDP literature is the importance of exploiting the problem structure to reduce the complexity of the learning and planning problems (Pineau, 2004b; Poupart, 2005). As detailed in the next chapter, the work presented in this thesis precisely attempts to transfer this insight into dialogue management.

3.2.3 Factored representations

In the previous pages, we modelled the system states and actions as atomic symbols. Such plain representations can unfortunately quickly lead to a combinatorial explosion of the state-action spaces. A more efficient alternative is to apply *factored* representations that decompose the state into distinct variables with possible conditional dependencies between one another. Similarly, action variables can also be split into distinct variables. For a MDP, the state will take the form of a set of variables $\mathbf{S}_t$, and the transition function $P(\mathbf{S}_t \mid \mathbf{S}_{t-1}, \mathbf{A}_{t-1})$ be represented as as a dynamic Bayesian network (Boutilier et al., 1999). The reward function can also be encoded as a collection of utility variables $\mathbf{R}_t$ connected to relevant sets of state and action variables. In such case, the total utility is defined as the sum of all utilities.

POMDPs can be factored in a similar way, with the inclusion of observation variables $\mathbf{O}_t$ connected to the state variables $\mathbf{S}_t$ through conditional dependencies $P(\mathbf{O}_t \mid \mathbf{S}_t)$. At time t, the observation variables $\mathbf{O}_t$ will be observed while the state variables $\mathbf{S}_t$ remain hidden. Figure 3.7 illustrates this factorisation. Both plain and factored (PO)MDPs form specific cases of dynamic

[8] A typical example of such action in dialogue is a clarification request about the user intention.

decision networks, as illustrated in the figure.

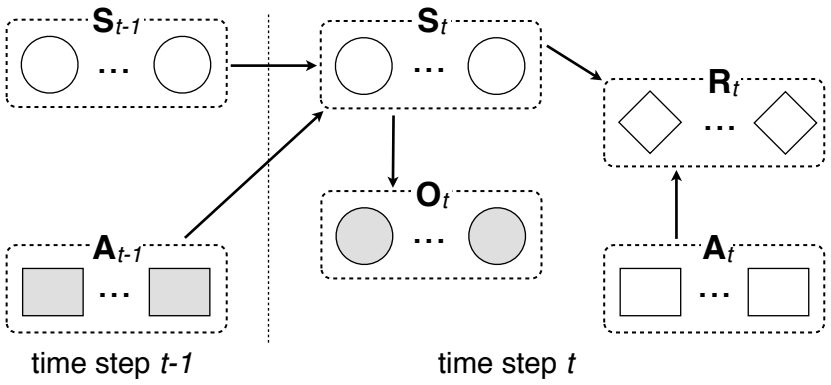


Figure 3.7: Factored representation of a POMDP with state variables **S**, action variables **A**, observations variables **O**, and reward variables **R**.

3.3 Application to dialogue management

We have now reviewed the key ideas of probabilistic reasoning and reinforcement learning, and are ready to explain how these ideas can be practically transferred to the dialogue management task. After a brief review of supervised learning approaches, we detail how dialogue can be modelled as a Markov Decision Process or Partially Observable Markov Decision Process, and survey the various optimisation techniques that have been developed to automatically optimise dialogue policies for such models based on real or simulated interaction data.

3.3.1 Supervised learning from Wizard-of-Oz data

The most straightforward use of statistical approaches to dialogue management is to learn dialogue policies in a supervised learning manner, based on collected Wizard-of-Oz data where dialogue management is remotely performed by a human expert. The resulting training data is a sequence $\{(s_i, a_i) : 1 \leq i \leq n\}$ of state-action pairs, where s_i is the state at time i and a_i the corresponding wizard action, which is assumed to reflect the best action to perform in this state. Most supervised learning approaches encode the dialogue state as a list of feature-value pairs, and the goal of the learning algorithm is to train a classifier $C : \mathcal{S} \rightarrow \mathcal{A}$ from state to actions that produces the best fit for this training data (modulo regularisation constraints), and will therefore "mimic" the decisions of the wizard in similar situations. Various classifiers can be used for this purpose, such as maximum likelihood classification (Hurtado et al., 2005), decision trees (Lane et al., 2004), Naive Bayes (Williams and Young, 2003), and logistic regression (Rieser and Lemon, 2006; Passonneau et al., 2012).

One important issue in supervised learning approaches is data sparsity, as only a fraction of the possible states can realistically be covered by the Wizard-of-Oz interactions. Several generalisation techniques can be employed to alleviate this problem. The simplest is to encode in a parametric form in which the parameters are associated with a smaller number of features extracted from

the state-action pair. The size of the feature set can be further reduced with feature selection (Passonneau et al., 2012). Yet another approach put forward by Hurtado et al. (2005) is to couple the classifier with a distance measure between states, thereby allowing the reuse of strategies learned from closely related states.

3.3.2 MDP-based optimisation of dialogue policies

Instead of learning a dialogue policy by imitating the behaviour of human experts, the dialogue manger can also learn by itself the best action to perform in each possible conversational situation via repeated interactions with a (real or simulated) user. Dialogue management can indeed be cast as a type of Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$ in which:

- The state space $\mathcal{S}$ corresponds the set of possible dialogue states, usually encoded as a list of feature-value pairs that capture relevant aspects of the current conversational context.
- The actions space $\mathcal{A}$ corresponds to the set of (verbal or extra-verbal) actions that can be executed by the system.
- The transition function T captures the "dynamics" of the conversation, and indicates how the dialogue state is expected to change as a result of the system actions (and in particular how the user is expected to respond).
- The reward function R expresses the objectives and costs of the application. A common reward function is to assign a high positive value for the successful completion of the task, a high negative value for a failure, and a small negative value for soliciting the user to repeat or clarify her/his intention.

A common misunderstanding should be clarified at this point. The representation of dialogue as a Markov Decision Process implies that the transition function only considers the previous state and action to predict the next state. It has occasionally been argued that this formalisation renders the decision-making process oblivious to non-local aspects of the dialogue history. This argument is however invalid, as the dialogue state is not limited to the mere representation of the last dialogue act, but may express any features related to the interaction context, including variables recording dialogue histories of arbitrary length, long-term user intentions, and so forth. It is ultimately up to the system designer to decide which features are deemed relevant to describe the current state of the dialogue.

Dialogue state representation

As for supervised learning methods, MDP-based reinforcement learning approaches to dialogue management encode the dialogue state in terms of feature-value pairs. The dialogue state is thus factored into a number of independent variables (one for each feature). Most early approaches adopted crude state representations with features limited to the status of the slots to fill and the last user utterance (Levin et al., 2000; Singh et al., 2000; Scheffler and Young, 2002). The voice-enabled email client described in Walker (2000) captured additional measures related to the overall task progress, history of previous system attempts, confidence thresholds and timing information. There has also been some work on the automatic identification of relevant state variables, using

methods from structure learning in decision networks (Paek and Chickering, 2006) and feature selection (Tetreault and Litman, 2006).

Henderson et al. (2008) are to our knowledge the first to explore the extension of reinforcement learning methods to large state spaces based on rich representations of the conversational context. Inspired by information state approaches to dialogue management, their state space captures detailed information such as complete history of dialogue acts and fine-grained representations of the task status, amounting to a total of 10^{386} possible states. Such rich state representations allow the dialogue manager to exploit much broader contextual knowledge in its decision-making. However, it also creates important challenges regarding action selection, as generalisation techniques are necessary to scale up the learning procedure to such large state spaces. '

Policy optimisation (and associated generalisation techniques)

For most dialogue domains, the reward is fixed in advance by the system designer and reflects the task objectives. It can also correspond to a performance metric such as PARADISE (Walker, 2000). The metric is defined in such case as a linear combination of quantitative measures whose weight is empirically estimated via multivariate linear regression from surveys of user satisfaction.[9]

The transition probabilities are however typically unknown. As described in Section 3.2.1, two families of approaches can be followed to optimise dialogue policies: model-based approaches first learn an explicit model of the MDP from interaction data and then extract a policy for this model, while model-free approaches directly estimate an action-value function $Q^*(s, a)$ from interactions, without explicit model of the state transitions.

While it is possible to follow a model-based strategy and learn explicit distributions for the transition probabilities (Singh et al., 2002; Tetreault and Litman, 2006), the majority of recent RL approaches to dialogue management have adopted model-free techniques. Due to the significant amounts of data necessary to reach convergence, it is often impossible to directly learn the value function from interactions with real users for most practical domains. A user simulator is instead used to provide unlimited supplies of interactions to the reinforcement learning algorithm. Several options are available to construct this user simulator. The first option is to design it manually based on specific assumptions about the user behaviour (Pietquin and Dutoit, 2006; Schatzmann et al., 2007a). It can also be constructed in a data-driven manner from existing corpora (Georgila et al., 2006).[10] In such case, user simulation can be interpreted as a way to expand the initial data set. The third available option is to exploit Wizard-of-Oz studies to tune the simulator parameters (Rieser and Lemon, 2010b). In addition to user modelling, the simulator should also integrate error modelling techniques in order to simulate various types of errors that may appear along the speech recognition and understanding pipeline (Schatzmann et al., 2007b; Thomson et al., 2012)

The key benefit of user simulation lies in the possibility to explore a large number of possible dialogue trajectories and error types. Such simulation is in particular of great use for prototyping dialogue policies and experimenting with alternative setups. However, simulated interactions are

[9] Quantitative measures of dialogue performance are typically classified in three categories: *task success* (e.g. ratio of successfully completed vs. failed tasks, κ agreement for slot-filling applications), *dialogue quality* measures (e.g. number of repetitions, barge-ins, ASR rejection prompts) and *dialogue efficiency* measures (e.g. number of utterances per dialogue, total elapsed time).

[10] The most important corpus of human-machine dialogue is the COMMUNICATOR dataset of telephone conversations in the domain of travel planning, with more than 180,000 utterances (90 hours of recording) transcribed and annotated with various understanding and dialogue-level information (Bennett and Rudnicky, 2002).

not as valuable as real interactions, and offer no guarantee of matching the behaviour and error patterns of actual users. The practice of evaluating the performance of dialogue policies based on the same simulator as the one used for training has also been put into question (Paek, 2006)

A key problem in reinforcement learning methods is the *curse of dimensionality*: the number of parameters grows exponentially with the size of the state and action spaces (Sutton and Barto, 1998). In order to scale the learning procedure to larger and more complex domains, factorisation and generalisation techniques are often necessary. An early example of this line of research is the work of Paek and Chickering (2006) on the use of graphical models in dialogue policy optimisation. The dialogue domain used in their experiments was a speech-enabled web browser. Their strategy was to explicitly represent the dialogue management task as a dynamic decision network and learn both the structure and parameters of this network from user simulations. The state space initially included all features that could be automatically logged from the interactions. Based on the simulated dialogues, the learning algorithm was able to automatically discover the subset of state variables that were relevant for decision-making as well as the transition probabilities between these variables. They also experimented with various Markov orders to analyse the impact of longer state histories on the system performance. After the estimation of the decision network, dynamic programming techniques were used to extract a dialogue policy that is optimal with respect to the learned models. Given the complexity of the optimisation, the dynamic programming solution was approximated via forward sampling (Kearns, 1999).

Henderson et al. (2008) demonstrated how to reduce the complexity of model-free learning techniques via function approximation. As the large size of their state-action space prevented the use of classical tabular representations for the $Q(s, a)$ function, they instead relied on function approximation to define the Q-values as a linear function of state features:

$$Q(s, a) = f(s)^T w_a \tag{3.20}$$

where $f(s)$ is a feature vector extracted from the state and w_a a weight vector for action a. The weight vectors were learned with the SARSA algorithm based on a fixed corpus (thereby avoiding the use of user simulators). To further refine the estimation of Q-values, they also combined estimates from both supervised and reinforcement learning in their final model.

Hierarchical abstraction is another way to reduce the search space of the optimisation process, as shown by Cuayáhuitl (2011). Instead of viewing action selection as a single monolithic policy, the selection is decomposed in their work into multiple levels, each responsible for a specific decision problem. This decomposition is formally expressed via an hierarchical extension of MDPs called Semi-Markov Decision Processes. Each level in the hierarchy is associated with its own subset of state and action variables. This modular approach allows a complex policy to be split into a sequence of simpler decisions. Such hierarchical formalisation is particularly natural for task-oriented dialogues, which are known to exhibit rich attentional and intentional structures, as notably argued in the seminal work of Grosz and Sidner (1986). It also bears similarities with the approach presented by Litman and Allen (1987) on dialogue understanding based on a plan structure.

Finally, it is worth noting that several researchers have attempted to combine the benefits of supervised and reinforcement learning methods by initialising a RL algorithm with a policy estimated via supervised methods (Williams and Young, 2003; Rieser and Lemon, 2006).

3.3.3 POMDP-based optimisation of dialogue policies

To capture the uncertainty associated with many state variables, dialogue can be explicitly modelled as a Partially Observable Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, Z \rangle$. Modelling a dialogue domain as a POMDP is similar in most respects to the MDP formalisation. The observations in $\mathcal{O}$ typically correspond to the possible N-best lists that can be generated by the speech recogniser and NLU modules, and can also include observations perceived via the other modalities.

Dialogue state representation

POMDP approaches express state uncertainty through the definition of a belief state b, which is a probability distribution $P(s)$ over possible states. After a system action a in belief state b followed by observation o, the belief state b is updated according to Equation (3.15), repeated here for convenience:

$$b' = P(s'|b, a, o) = \eta \; P(o|s') \sum_{s} P(s'|s, a) \; b(s) \tag{3.15}$$

Belief update requires the specification of two probabilistic models: the observation model $P(o|s')$ and the transition model $P(s'|s, a)$. Many POMDP approaches to dialogue management factor the state s into (at least) three distinct variables $s = \langle a_u, i_u, c \rangle$, where a_u is the last user dialogue act, i_u the current user intention(s), and c the interaction context.[11] Assuming that the observation o only depends on the last user act a_u, and that a_u depends on both the user intention i_u and the last system action a_m, Equation (3.15) is then rewritten as:

$$\begin{aligned} b' &= P(a'_u, i'_u, c' \,|\, b, a_m, o) && (3.21) \\ &= \eta \; P(o \,|\, a'_u) \; P(a'_u \,|\, i'_u, a_m) \sum_{i_u, c'} P(i'_u \,|\, i_u, a_m, c') P(c') \; b(i_u) && (3.22) \end{aligned}$$

The transition model is decomposed in this factorisation into two distinct distributions:

1. The distribution $P(a'_u \,|\, i'_u, a_m)$ is called the *user action model* and defines the probability of a particular user action given her/his underlying intention and the last system act. It expresses the likelihood of the user action a'_u following the system action a_m and the compatibility of a'_u with the user intention i'_u (Young et al., 2010).

2. The distribution $P(i'_u \,|\, i_u, a_m, c')$ is the *user goal model* and captures how the user intention is likely to change as a result of the context and system actions.

These two distributions are usually derived from collected interaction data. A graphical illustration of this state factorisation is shown in Figure 3.8.

The observation model $P(o \,|\, a'_u)$ is often rewritten as $P(\tilde{a}_u)$, the dialogue act probability in the N-best list provided by the speech recognition and semantic parsing modules (cf. Section 2.2.2), based on the following approximation:

$$P(o \,|\, a'_u) = \frac{P(a'_u \,|\, o) \; P(o)}{P(a'_u)} \approx P(a'_u \,|\, o) = P(\tilde{a}_u) \tag{3.23}$$

[11] Some approaches also define a specific variable for the dialogue history (Young et al., 2010).

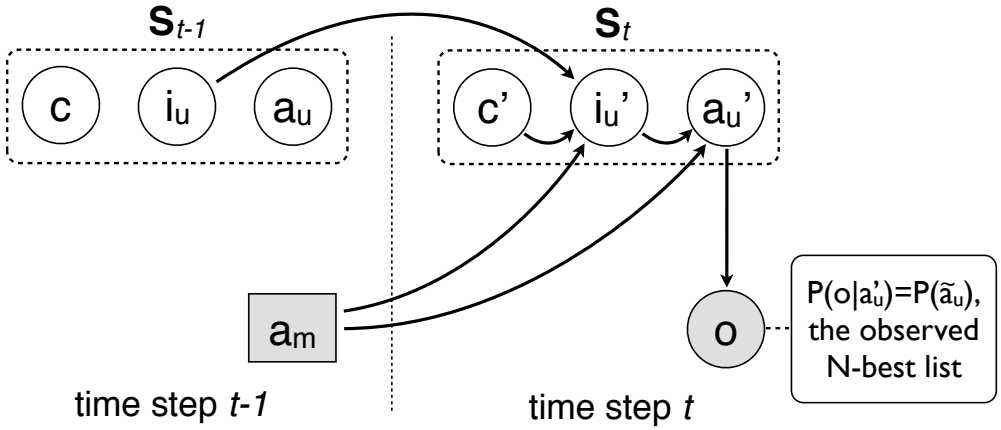


Figure 3.8: Common factorisation of the state space for a POMDP-based dialogue system, where c represents the dialogue context, i_u the user intention(s), a_u the user dialogue act, a_m the system act, and o the observed N-best list hypotheses. The representation omits the conditional dependencies and probability distribution for the variable c', which are contingent on the particular interaction context in place for the domain.

The approximation rests on the assumption of uniform distribution for the user dialogue acts and observation in the absence of further evidence. Since the probabilities $P(\tilde{a}_u)$ are provided at runtime by the ASR and NLU modules, the approximation does no require an explicit specification of the observation set $\mathcal{O}$ and observation model Z at design time. This modelling choice has the advantage of circumventing the statistical estimation of the observation model, a difficult problem since the number of possible N-best lists is theoretically infinite. A few researchers have however attempted to estimate explicit observation models for small dialogue domains. Williams et al. (2008) investigated how to integrate observations that include continuous-valued ASR confidence scores into a classical POMDP framework and devised ad-hoc density functions for this purpose. Chinaei et al. (2012) showed how an observation model could be empirically estimated from interaction data with a bag-of-words approach.

In the seminal work of Roy et al. (2000) that first introduced the POMDP framework to dialogue management, the state is represented by a single variable expressing the user intention, and hand-crafted models were used for the belief update. Zhang et al. (2001) extended the previous approach by introducing a factored state representation based on Bayesian Networks, where the state includes both the user intention and the system state. Williams et al. (2005); Young et al. (2010) further refined this factorisation by decomposing the dialogue state into three distinct variables that respectively represent the last user dialogue act, the user intention and the dialogue history. The related work of Thomson and Young (2010) used Bayesian networks to encode fine-grained dependencies between the various slots expressed in the user intention. Bui et al. (2010) added to this factorisation a specific variable for the user's affective state. Substantial work has also been devoted to the inclusion of non-verbal observations and environmental factors into the dialogue state. In the human-robot interaction domain, Prodanov and Drygajlo (2003); Hong et al. (2007) have notably applied Bayesian networks for inferring the underlying user intention based on observations arising from both verbal and non-verbal sources.

Policy optimisation

POMDP solutions methods can in theory be applied to extract the dialogue policy from any given model specification, as shown by Williams and Young (2007); Williams et al. (2008). Such strategy is however only suitable for relatively small action-state spaces and require the specification of an explicit observation model to extract the α-vectors corresponding to the optimal policy (Shani et al., 2013). Most recent POMDP approaches have instead focused on the use of reinforcement learning to derive a dialogue policy from interactions with a user simulator (Young et al., 2010; Thomson and Young, 2010; Daubigney et al., 2012b).

The need for effective generalisation techniques is heightened in reinforcement learning for POMDPs, as dialogue policies defined in partially observable environments must be defined in a high-dimensional, continuous belief state space. Approximating the $Q^*(b, a)$ function is thus crucial. One useful approximation method is to reduce the full belief state to a simpler representation such as the "summary state" described by Williams and Young (2005). In addition, the estimation of the action-value function can be further simplified by relying on techniques such as grid-based discretisations (Young et al., 2010) and linear function approximation (Thomson and Young, 2010; Daubigney et al., 2012b). Finally, non-parametric methods based on Gaussian Processes have recently been proposed (Gašić et al., 2011).

Reinforcement learning is considerably more difficult for POMDPs than for MDPs due to the partial observability of the state. Png and Pineau (2011) showed how model-based reinforcement learning can be cast in a Bayesian framework. The key idea of Bayesian reinforcement learning is to maintain an explicit probability distribution over the POMDP parameters. This distribution is then gradually refined as more data is observed. This refinement is done using Bayesian inference, starting with an initial prior. At runtime, the parameter distribution is applied to plan the optimal action, taking into account every source of uncertainty – that is, state uncertainty, stochastic action effects, and uncertainty over the parameters. Our own work on model-based reinforcement learning also follows that line of work, as shall be explained in Chapter 6.

Most POMDP-based approaches to dialogue management assume that the reward model can be encoded in advance by the system designer, with a few notable exceptions. Atrash and Pineau (2009) describe a Bayesian approach to estimate a reward model based on gold standard actions provided by an oracle. Boularias et al. (2010); Chinaei and Chaib-draa (2012) present alternative approaches based on inverse reinforcement learning (IRL) for POMDPs. Their main idea was to exploit Wizard-of-Oz data for a voice-enabled intelligent wheelchair to automatically infer a reward model. This task of inferring a reward model from expert demonstrations is a prototypical instance of *inverse reinforcement learning*: the agent observes how an expert performs the task and must find the hidden reward model that best explains this behaviour. Inverse reinforcement learning in partially observable domains is however difficult to scale beyond small domains due to the complexity of the optimisation problem (Choi and Kim, 2011).

3.4 Summary

We have exposed down in this chapter the foundations of probabilistic modelling applied to dialogue, and have reviewed a range of theoretical concepts related to graphical models, reinforcement learning in both fully and partially observable domains, as well as the practical exploitation of these

techniques to dialogue management.

The first section of this chapter focused on the use of efficient representations of probability and utility models. We described how directed graphical models could capture various probability and utility distributions. We reviewed in particular the main properties of Bayesian networks, dynamic Bayesian networks, and dynamic decision networks, and described the most important algorithms for inference and parameter estimation that have been tailored for these graphical models. The appeal of graphical models for the representation of complex stochastic phenomena is elegantly described by Jordan (1998, p. 1):

> "Graphical models, a marriage between probability theory and graph theory, provide a natural tool for dealing with two problems that occur throughout applied mathematics and engineering – uncertainty and complexity. In particular, they play an increasingly important role in the design and analysis of machine learning algorithms. Fundamental to the idea of a graphical model is the notion of modularity: a complex system is built by combining simpler parts. Probability theory serves as the glue whereby the parts are combined, ensuring that the system as a whole is consistent and providing ways to interface models to data. Graph theory provides both an intuitively appealing interface by which humans can model highly interacting sets of variables and a data structure that lends itself naturally to the design of efficient general-purpose algorithms."

The second section laid down the core concepts and techniques in the field of reinforcement learning. We described the formal notion of a Markov Decision Process (MDP), composed of a set of states, a set of actions, a transition function describing temporal relations between states, and a reward function encoding the utilities of particular actions. We explained how MDPs can be extended to capture partial observability (leading to POMDPs), and how policies can be optimised for both MDPs and POMDPs using model-based and model-free techniques.

The final section translated these formal representations and optimisation techniques to the practical problem of dialogue management. Three learning strategies were distinguished: supervised learning, reinforcement learning with MDPs, and reinforcement learning with POMDPs. We surveyed a variety of approaches that differ along dimensions such as the representation of the dialogue state, the learning algorithm employed for the optimisation, the type and structure of the models that are to be estimated, and the source of data samples that is employed for this estimation. We discussed the respective merits and limitations of these dialogue optimisation strategies, and stressed in particular the importance of factorisation and generalisation methods to handle the complexity of real-world dialogue domains. The next chapter will now present in detail our own approach to this prominent problem.

Chapter 4

Probabilistic rules

The previous chapter fleshed out how dialogue could be represented as a stochastic process and described the benefits of using graphical models to efficiently encode the probability and utility models employed in dialogue management. Plain graphical models must however face non-trivial scalability issues when applied to dialogue domains associated with rich conversational contexts. The number of parameters necessary for state update and action selection can indeed increase rapidly with the complexity of the domain models. Alas, only small quantities of genuine training data are available in most dialogue domains, and usually cover only a small fraction of the state-action space of interest.

To address this discrepancy between the size of the parameter space and the amount of data available to estimate them, we introduce in this chapter a new approach to probabilistic dialogue modelling, based on the notion of *probabilistic rules*. Probabilistic rules are structured mappings between conditions and effects, and function as *high-level templates* for the construction of a dynamic decision network. The key advantage of this structured modelling approach is the drastic reduction of the number of parameters compared to traditional representations. We also argue that these expressive representations are particularly well suited to encode the probability and utility models of dialogue domains, where substantial amounts of expert knowledge can often be leveraged to structure the relationships between variables.

The chapter is divided in six sections. Section 4.1 exposes in general terms how structural assumptions can be applied to reduce the size and complexity of probabilistic models. Section 4.2 defines the formalism of probabilistic rules and its main theoretical properties. These definitions are then connected in Section 4.3 to the graphical models described in the previous chapter by showing how the rules are practically instantiated in the Bayesian network representing the dialogue state. Section 4.4 explains how this instantiation procedure is incorporated in the processing workflow for updating the dialogue state and selecting the system actions. Finally, Section 4.5 addresses some advanced modelling questions and Section 4.6 relates the approach to previous work.

4.1 Structural leverage

The starting point of our approach is the observation that the probability and utility models used in dialogue management often exhibit a fair amount of *internal structure*. We have already discussed in the previous chapter one simple instance of this internal structure, namely factored representations based on conditional independences. However, the internal structure of dialogue domains does not

limit itself to these basic independence assumptions, and much can be gained by exploiting other types of structural properties, as shall be argued in the next pages.

Latent variables

The number of parameters required to estimate the distributions of a graphical model can often be reduced by introducing *latent variables* (i.e. unobserved or hidden variables) that act as intermediaries between the source and target variables. Indeed, many application domains are often best explained by the combination of a small number of distinct factors or influences, each encoded by a separate random variable and associated with a subset of input and output variables. This layer of latent variables is usually never observed directly but contribute to structuring the model.[1] In the particular case of medical diagnosis, the relations between predisposing factors and observed symptoms are for instance best described by postulating an intermediary layer of variables – possible diseases – that mediates between the predisposing factors and the observed symptoms. Figure 4.1 illustrates how latent variables can be exploited to provide an additional layer of abstraction within a graphical model.

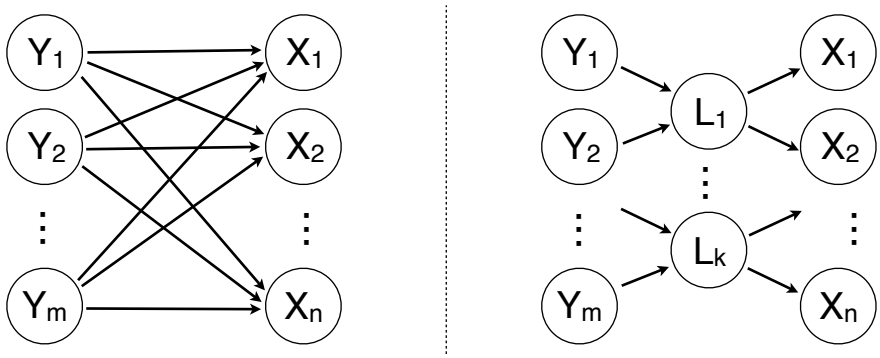


Figure 4.1: Comparison between a model that directly maps variables $\mathbf{Y}$ to $\mathbf{X}$ (left side) and one relying on latent variables $\mathbf{L}$ to serve as intermediaries (right side).

Dialogue models can benefit from the inclusion of such latent variables. The transition function can for instance be modelled in terms of a limited number of latent variables, each responsible for capturing specific aspects of the interaction dynamics. We shall see in the forthcoming sections that probabilistic rules precisely operate as latent variables when instantiated in the dialogue state.

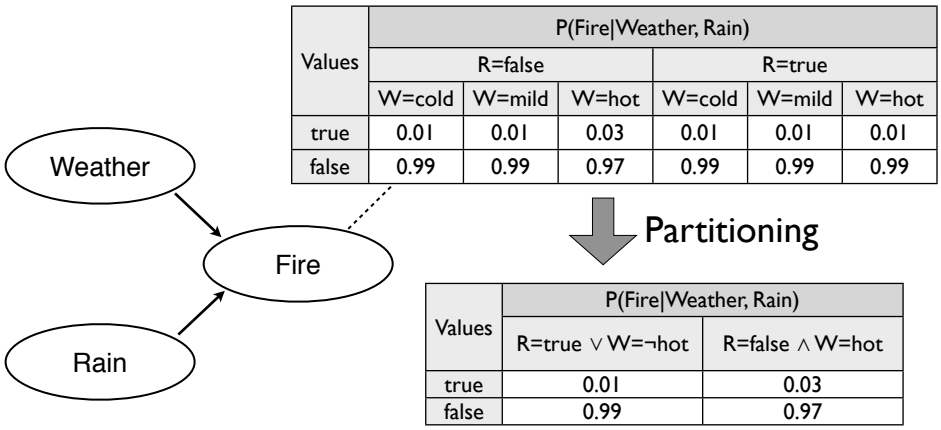
Partitioning

A random variable X with parent variables $Y_1, ...Y_m$ must specify a separate probability distribution for every possible assignment of values for the parent variables. In other words, the number of parameters required to specify the distribution $P(X \mid Y_1, ..., Y_m)$ is exponential in the number of parents m. Fortunately, the values of these parents variables can be grouped into *partitions* yielding similar outcomes for X. One can therefore directly define the conditional probability distributions on these groups rather than on the full enumeration of combined values for the parent

[1] The construction of layered computational models is one of the most active research topic in artificial intelligence and machine learning, and form in particular the foundations of deep learning approaches (Bengio, 2009).

variables. Partitioning is an example of *abstraction mechanism* which can be used to reduce the model complexity and improve its ability to generalise to unseen examples.

Figure 4.2 illustrates such partitioning operation for the conditional probability distribution $P(Fire \mid Weather, Rain)$. The space of possible values for the parent variables is defined in this example as $Val(Weather) \times Val(Rain)$ and contains 6 possible elements. We can observe that this space can be split in two partitions: $Rain = true \vee Weather \neq hot$ and $Rain = false \wedge Weather = hot$. This partitioning allows a significant reduction of the number of parameters required for the conditional probability distribution. It should be noted that grouping value assignments into partitions corresponds to a modelling choice and can degrade the model accuracy if the partitions do not reflect actual similarities in the predicted outcomes.



Values	P(Fire|Weather, Rain)					
	R=false			R=true		
	W=cold	W=mild	W=hot	W=cold	W=mild	W=hot
true	0.01	0.01	0.03	0.01	0.01	0.01
false	0.99	0.99	0.97	0.99	0.99	0.99

Values	P(Fire|Weather, Rain)	
	R=true ∨ W=¬hot	R=false ∧ W=hot
true	0.01	0.03
false	0.99	0.97

Figure 4.2: Partitioning for the conditional probability distribution $P(Fire \mid Weather, Rain)$.

Partitions must be both exhaustive (each combination of values for the parent variables must belong to one partition) and mutually exclusive (a combination of values can only belong to one partition). As we can observe from the example, partitions can often be concisely expressed via logical conditions on the variable values. A given assignment of values is then grouped in a partition if it satisfies the condition associated with it.

Quantification

Many dialogue domains are composed of objects or entities related to one another. These domains are often difficult to directly encode by a fixed set of random variables, as the number of entities and relations may vary over time. Examples of relational structures include:

- Collections of physical objects in a visual scene, each described by specific features (colour, shape) and relations with other objects (e.g. spatial relations),
- Indoor environments topologically structured in rooms and spaces in which to navigate.
- Stacks of tasks to complete by the agent, each task being possibly connected to other tasks via precedence or inclusion relationships.
- Linguistic entities employed in the dialogue acts of the conversational partners, linked with one another through multiple syntactic, semantic, referential or pragmatic relations.

First-order logic provides an excellent basis for representing and manipulating such relational structures, as it offers a principled language for (1) referring to objects connected with one another through functions and relations and (2) describing their properties in a concise way through the use of universal and existential quantifiers.[2]

Graphical models represent relational domains by instantiating one random variable for every possible grounding of the functions and predicates defined for the domain for a collection of objects.[3] A domain with two objects o_1 and o_2 and a relation $leftOf(x, y)$ will for instance generate the four groundings $leftOf(o_1, o_2)$, $leftOf(o_2, o_1)$ $leftOf(o_1, o_1)$ and $leftOf(o_2, o_2)$. The definition of probability and utility distributions that can handle the relational semantics of such representation is however problematic. In particular, generic properties and constraints such as $\forall x, \neg leftOf(x, x)$ and $\forall x, y, z, leftOf(x, y) \wedge leftOf(y, z) \Rightarrow leftOf(x, z)$ can be difficult to enforce at a global level, since classical probabilistic models offer no direct support for quantifiers. Their expressive power is indeed intrinsically limited to propositional logic.

The unification of first-order logic and probability theory has spanned a new research area called *statistical relational learning* (Getoor and Taskar, 2007). A common trait of most approaches to statistical relational learning is the definition of a logic-based description language which is employed as a template to generate classical probabilistic models given a set of constants. The introduction of quantifiers provide an abstraction mechanism to reduce the complexity of probability and utility models by describing constraints or relations that hold for all possible groundings of a given formula and can therefore apply to large sets of random variables.

4.2 Formalisation

We now outline a generic description framework for expressing the various types of internal structure we have just detailed. This description framework revolves around the notion of *probabilistic rules*.The framework was originally presented in Lison (2012d,a).

The key idea is to represent distributions with the help of *if ... then ... else* control structures, based on the following skeleton:

if (condition 1 holds) **then**
    Distribution 1 over possible effects
else if (condition 2 holds) **then**
    Distribution 2 over possible effects
...
else
    Distribution n over possible effects

Each *if ... then* branch specifies both a *condition* on particular state variables and an associated distribution over possible *effects*. The *if ... then ... else* structure is read in sequential order, as in programming languages, until a satisfied condition is found, which causes the activation of the

[2] We shall not cover in this thesis the mathematical foundations of first-order logic, but the interested reader is invited to refer to e.g. Gamut (1991) for a formal overview of the logical concepts mentioned throughout this thesis.

[3] Such operation is akin to *propositionalisation* in the terminology of first-order logic.

corresponding probabilistic effects.

We first present how probabilistic rules can express conditional probability distributions in terms of structured mappings between input and output variables. We then show how to generalise the formalism to utility distributions and extend it with quantification mechanisms.

A terminological note is here in order: we shall use the term *probabilistic rules* as an umbrella term that covers all types of rules in this thesis, while *probability rules* will only refer to rules expressing probability distributions over effects, and *utility rules* to rules expressing utility distributions.

4.2.1 Probability rules

Probability rules take the form of *if... then ... else* control structures and map a list of conditions on input variables to probabilistic effects on output variables. More formally, a rule is expressed as an ordered list $\langle br_1, ...br_n \rangle$, where each branch br_i is a pair $(c_i, P(E_i))$, c_i is a condition and $P(E_i)$ an associated distribution over possible effects. The distribution $P(E_i)$ is a categorical distribution with possible effects $Val(E_i) = \{e_{(i,1)}, ...e_{(i,m_i)}\}$, where m_i is the number of alternative effects in $P(E_i)$. Each effect $e_{(i,j)} \in Val(E_i)$ has a particular probability denoted $p_{(i,j)}$.

Given these elements, a basic probability rule reads as such:

$$
\begin{aligned}
&\textbf{if } (c_1) \textbf{ then} \\
&\quad \begin{cases} P(E_1 = e_{(1,1)}) = p_{(1,1)} \\ \dots \\ P(E_1 = e_{(1,m_1)}) = p_{(1,m_1)} \end{cases} \\
&\textbf{else if } (c_2) \textbf{ then} \\
&\quad \begin{cases} P(E_2 = e_{(2,1)}) = p_{(2,1)} \\ \dots \\ P(E_2 = e_{(2,m_2)}) = p_{(2,m_2)} \end{cases} \\
&\dots \\
&\textbf{else} \\
&\quad \begin{cases} P(E_n = e_{(n,1)}) = p_{(n,1)} \\ \dots \\ P(E_n = e_{(n,m_n)}) = p_{(n,m_n)} \end{cases}
\end{aligned}
\tag{4.1}
$$

In the rest of this thesis, we will often use $P(e_{(i,j)})$ as notational convenience for $P(E_i = e_{(i,j)})$.

Conditions

The conditions c_i are expressed as logical formulae grounded in a subset of random variables defined in the dialogue state. This subset of state variables are the *input variables* of the rule, which we shall denote as $I_1, ...I_k$. Conditions can be arbitrarily complex logical formulae connected by conjunction, disjunction and negation, and (as we shall see in Section 4.2.3) can also include universally quantified variables. The examples $(Rain = true \vee Weather \neq hot)$ and

$(Rain = false \wedge Weather = hot)$ in Figure 4.2 are instances of valid conditions on the two input variables $Rain$ and $Weather$.

Given that a rule is defined through a *if ... then ... else* control structure, the partitioning is guaranteed by construction to be exhaustive and mutually exclusive (only one branch will be followed). When provided with an assignment of values on the input variables, the conditions are tested in sequential order until one is satisfied. When no terminating **else** block is explicitly specified at the end of a rule, the framework assumes a final **else** block associated with a void effect to ensure that the partitioning is exhaustive. The last condition c_n is thus guaranteed to be always trivially satisfied irrespective of the input variable values.

The conditions on the input variables offer a compact partitioning of the state space to mitigate the dimensionality curse. Without this partitioning in alternative conditions, a rule ranging over input variables $I_1, ...I_k$ each containing q possible values would need to enumerate q^k possible assignments. Partitioning this space reduces this number to n mutually exclusive partitions, where n corresponds to the number of conditions for the rule and is usually small.

Effects

Associated to each condition c_i stands a collection of mutually exclusive effects $e_i^{1...m_i}$. Each effect $e_{(i,j)}$ defines a specific assignment of values for a set of variables called the *output variables* of the rule. An effect is defined as a conjunction of (variable,value) pairs $O_1 = o_1 ... \wedge O_l = o_l$ where $O_1, ...O_l$ are the output variables (which may already exist or yet to be created) and $o_1, ...o_l$ the corresponding values for these variables. In the partitioning example from Figure 4.2, the output variable is unique and corresponds to $Fire$. We shall however encounter examples of rules with more than one output variable.

Effects can be void – that is, represent an empty assignment. In such case, the effect does not lead to any change in the distribution of the output variables for the rule.

Probabilities

Each effect $e_{(i,j)}$ is assigned with a probability $p_{(i,j)} = P(E_i = e_{(i,j)})$ that must satisfy the usual probability axioms $p_{(i,j)} \geq 0 \ \forall i, j$ and $\sum_{j=1}^{m_i} p_{(i,j)} = 1 \ \forall i$. The probabilities can be either fixed by hand or correspond to parameters to estimate from data. Chapters 5 and 6 detail how Dirichlet distributions can be exploited to estimate probability parameters.

Example

Rule r_1 illustrates a simple example of probability rule:

$$
\begin{aligned}
r_1 : \quad & \textbf{if } (Rain = false \wedge Weather = hot) \textbf{ then} \\
& \begin{cases} P(Fire = true) = 0.03 \\ P(Fire = false) = 0.97 \end{cases} \\
& \textbf{else} \\
& \begin{cases} P(Fire = true) = 0.01 \\ P(Fire = false) = 0.99 \end{cases}
\end{aligned}
$$

Rule r_1 has two input variables: $Rain$ and $Weather$ as well as one output variable $Fire$. The rule specifies that the probability of a fire is 0.03 in case of no rain and a hot weather and 0.01 in all other cases. The rule structure enables the conditional probability distribution for $Fire$ to be specified with only four probabilities in comparison to twelve for the original CPD (Figure 4.2).

4.2.2 Utility rules

The rule-based formalism we have outlined can also be used to express utility distributions with only minor notational changes. Utility rules essentially retain the same form as probability rules, with one important exception, namely that the probabilistic effects are replaced by utility distributions over particular assignments of decision variables.

Formally, a utility rule is an ordered list $\langle br_1, ...br_n \rangle$, where each branch br_i is a pair (c_i, U_i) where c_i is a condition and U_i an associated utility distribution over possible assignments of decision variables. The utility distribution U_i specifies a set of possible decisions $d_{(i,1)}, ...d_{(i,m_i)}$. Each decision $d_{(i,j)}$ has a particular utility value denoted $u_{(i,j)}$. Utility rules can be expressed in the following manner:

$$
\begin{aligned}
&\textbf{if } (c_1) \textbf{ then} \\
&\quad \begin{cases} U_1(d_{(1,1)}) = u_{(1,1)} \\ ... \\ U_1(d_{(1,m_1)}) = u_{(1,m_1)} \end{cases} \\
&\textbf{else if } (c_2) \textbf{ then} \\
&\quad \begin{cases} U_2(d_{(2,1)}) = u_{(2,1)} \\ ... \\ U_2(d_{(2,m_2)}) = u_{(2,m_2)} \end{cases} \\
&... \\
&\textbf{else} \\
&\quad \begin{cases} U_n(d_{(n,1)}) = u_{(n,1)} \\ ... \\ U_n(d_{(n,m_n)}) = u_{(n,m_n)} \end{cases}
\end{aligned}
\tag{4.2}
$$

A utility rule assigns utility values to particular system decisions depending on conditions on the state variables. As for probability rules, the conditions c_i are defined as arbitrary logical formulae on input variables $I_1, ...I_k$. The decisions $d_{(i,j)}$ are assignments $A_1 = a_1 ... \wedge A_l = a_l$ where the variables $A_1, ..A_l$ are decision variables and $a_1, ...a_l$ possible values for these variables. The utility values $u_{(i,j)}$ are real numbers (which may be positive or negative).

Although most utility rules only include one single decision variable, the possibility to integrate multiple decision variables is helpful in domains where the system can execute multiple actions in parallel. Such situations can arise in human-robot interaction and multi-modal applications, as the system can communicate through both verbal and non-verbal channels and is often able to perform physical actions in addition to communicative acts.

Example

Rule r_2 provides a simple example of utility rule:

$$
\begin{aligned}
r_2 : \ & \textbf{if } (Fire = true) \textbf{ then} \\
& \quad \begin{cases} U(Tanker = drop\text{-}water) = 5 \\ U(Tanker = wait) = -5 \end{cases} \\
& \textbf{else} \\
& \quad \begin{cases} U(Tanker = drop\text{-}water) = -1 \\ U(Tanker = wait) = 0 \end{cases}
\end{aligned}
$$

Rule r_2 stipulates the respective utilities of the two possible utility values for the decision variable $Tanker$ depending on the variable $Fire$.

4.2.3 Quantification

Quantification is a powerful mechanism to abstract over particular relational aspects of the domain structure. Logical variables can be included in the specification of both the conditions and effects of a given rule, and are universally quantified on top of the rule.[4] A rule containing the quantified variables $y_1 ... y_p$ in its conditions and/or effects is therefore formalised as:

$$
\begin{aligned}
& \forall \mathbf{y} = y_1, y_2, ... y_p : \\
& \qquad \textbf{if } (c_1(\mathbf{y})) \textbf{ then} \\
& \qquad\quad \begin{cases} P(E_1 = e_{(1,1)}(\mathbf{y})) = p_{(1,1)} \\ ... \\ P(E_1 = e_{(1,m_1)}(\mathbf{y})) = p_{(1,m_1)} \end{cases} \\
& \qquad \textbf{else if } (c_2(\mathbf{y})) \textbf{ then} \\
& \qquad\quad \begin{cases} P(E_2 = e_{(2,1)}(\mathbf{y})) = p_{(2,1)} \\ ... \\ P(E_2 = e_{(2,m_2)}(\mathbf{y})) = p_{(2,m_2)} \end{cases} \\
& \qquad ... \\
& \qquad \textbf{else} \\
& \qquad\quad \begin{cases} P(E_n = e_{(n,1)}(\mathbf{y})) = p_{(n,1)} \\ ... \\ P(E_n = e_{(n,m_n)}(\mathbf{y})) = p_{(n,m_n)} \end{cases}
\end{aligned}
\tag{4.3}
$$

The formalisation allows specific elements $y_1, ... y_p$ in the conditions and effects to be *under-*

[4] These variables are variables in the sense of first-order logic, and are not to be confused with the random variables of the probabilistic model.

specified. The mapping between conditions and effects specified by the rule holds for every possible assignment of the underspecified variables. Based on this quantification mechanism, probabilistic rules can cover large portions of the state space in a highly compact manner, based on a reduced number of parameters. One of the key advantages of such representation is that it allows for powerful forms of *parameter sharing*, as the effect probabilities $p_{(i,j)}$ in the above rule are made independent of the various possible instantiations of the variables $y_1, ...y_p$. Quantification also applies to utility rules in the same manner.

Example

Rule r_3 provides a simple example of probability rule including a quantified variable:

$$
\begin{aligned}
r_3 : \; & \forall y : \\
& \textbf{if } (shape(y) = sphere) \textbf{ then} \\
& \quad \begin{cases} P(graspable(y) = true) = 0.9 \\ P(graspable(y) = false) = 0.1 \end{cases} \\
& \textbf{else if } (shape(y) = cone) \textbf{ then} \\
& \quad \begin{cases} P(graspable(y) = true) = 0.2 \\ P(graspable(y) = false) = 0.8 \end{cases}
\end{aligned}
$$

Rule r_3 specifies how the graspability of a given object y depends on its shape (a sphere being easier to grasp than a cone). Similarly, rule r_4 defines the utility of a grasping action depending on the task and object graspability:

$$
\begin{aligned}
r_4 : \; & \forall y : \\
& \textbf{if } (task = grasp(y) \wedge graspable(y) = true) \textbf{ then} \\
& \quad \begin{cases} U(a_m = grasp(y)) = 2 \end{cases} \\
& \textbf{else} \\
& \quad \begin{cases} U(a_m = grasp(y)) = -2 \end{cases}
\end{aligned}
$$

Rule r_4 associates an utility of 2 to the action of grasping an object y when it corresponds to the task and is feasible. Grasping the object in any other case results in a negative utility of -2.

4.3 Rule instantiation

We represent the dialogue state as a Bayesian network over state variables, in line with other dialogue management approaches such as Thomson and Young (2010); Bui et al. (2010). Rules are then applied at runtime on this dialogue state. The instantiation is performed by creating a distinct node for every rule to apply. These rule nodes are essentially latent variables that serve as intermediaries between input and output variables. Albeit the presence of these rules is never directly observed, they help structuring the relations between variables and enable the system designer to decompose complex probability and utility models into smaller parts.

We describe below the instantiation procedure for each type of rule. For the sake of clarity, we shall first limit our discussion to rules without quantifiers, and then demonstrate how quantifiers can be accounted for in the instantiation process.

4.3.1 Probability rules

Let $\mathcal{B}$ be the Bayesian network representing the current dialogue state, and $\mathcal{R}$ a set of rules to apply to this dialogue state. For each rule $r \in \mathcal{R}$, a distinct chance node is created.[5] This chance node represents a random variable defined on the possible effects of the rule. The node is conditionally dependent on the input variables of the rule (i.e. the set of all variables that are mentioned in the rule conditions), and is also connected via outgoing edges to its output variables (i.e. the set of all variables that are mentioned in the rule effects).

Figure 4.3 illustrates this construction process on a constructed example composed of the two rules r_5 and r_6. To simplify the rule representation, we shall usually omit the explicit specification of the probability for the empty effect in the effect distributions. The remaining probability mass in the rules is thus by default assigned to the empty effect.

The two rules r_5 and r_6 are applied on the state variables A, B, C and D. The application of the two rules results in an update of the variable A and the creation of a new variable E. The nodes corresponding to the output variables are by convention denoted with a prime to distinguish them from the input nodes.

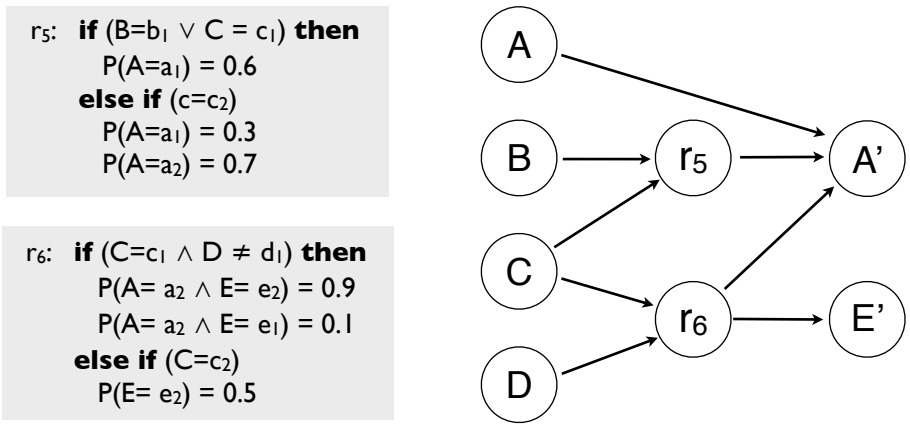


Figure 4.3: Example of instantiation for the two probability rules r_5 and r_6.

The random variable represented by the node r_5 has three possible values, reflecting the effects described in the rule: $Val(r_5) = \{\{A=a_1\}, \{A=a_2\}, \{\cdot\}\}$, where $\{\cdot\}$ denotes the empty effect. Similarly, the random variable r_5 has four alternative effects: $Val(r_6) = \{\{A=a_2 \wedge E=e_2\}, \{A=a_2 \wedge E=e_1\}, \{E=e_2\}, \{\cdot\}\}$.

We shall also adopt the following terminology to denote the probability distributions created through the instantiation procedure:

[5] The original instantiation algorithm presented in (Lison, 2012d) included two separate nodes: one for the rule condition and one for the effect. The formalism was later simplified to one single node.

- The conditional probability distribution associated with rule nodes such as r_5 and r_6 given their inputs is a *rule distribution*.
- The conditional probability distribution associated with output variables such A' and E' given the rule nodes that determines is an *output distribution*.

Rule distributions

The rule distributions directly reflect the rule semantics. Formally, the conditional probability distribution of a rule node r given its input variables $I_1, ...I_k$ is defined as:

$$P(r=e \mid I_1=i_1, ...I_k=i_k) = P(E_i = e) \quad (4.4)$$
$$\text{where } i = \min_i(c_i \text{ is satisfied with } I_1=i_1 \wedge ...I_k=i_k)$$

Formally speaking, a condition c_i is said to be satisfied iff the input assignment logically entails that the condition is true, that is :$(I_1 = i_1 \wedge ...I_k = i_k) \vdash c_i$. The rule conditions are checked in sequential order until one condition is found to be satisfied. The rule distribution is then simply determined as the effect distribution $P(E_i = e)$ associated with the first satisfied condition c_i. As the last condition c_n corresponds to the final **else** block and is therefore always trivially true, there will always be at least one satisfied condition.

As an example, the rule distribution $P(r_5 \mid B = b_1, C = c_1)$ for the node r_5 in Figure 4.3 is defined as:

- $P(r_5 = \{A=a_1\} \mid B=b_1, C=c_1) = 0.6$
- $P(r_5 = \{\cdot\} \mid B=b_1, C=c_1) = 0.4$

Similarly, the distribution $P(r_6 \mid C=c_1, D=d_1)$ is a distribution with the empty effect $\{\cdot\}$ assigned to a probability 1.

Output distributions

An output node X' is conditionally dependent on all the rule nodes that refer to the variable X in their effects. In addition, output nodes that correspond to the updated version of existing nodes (such as A' in the example of Figure 4.3) also include a conditional dependence on these existing nodes. The output distribution is a reflection of the combination of effects specified in the parent rules. The conditional probability distribution $P(X'|r_1=e_1, ...r_n=e_n)$ for an output variable X' with n incoming rule nodes is defined in the following manner:

$$P(X'=x' \mid r_1=e_1, ...r_n=e_n) = \begin{cases} \frac{\sum_{v \in \mathbf{e}(X)} \mathbf{1}(x'=v)}{|\mathbf{e}(X)|} & \text{if } \mathbf{e}(X) \neq \emptyset \\ \mathbf{1}(x' = None) & \text{otherwise} \end{cases} \quad (4.5)$$

where the following notation is used:

- $\mathbf{e}$ is the conjunction of all effects, i.e. $\mathbf{e} = e_1 \wedge ...e_n$. This conjunction can include more than one assigned value for a particular variable.

- $\mathbf{e}(X)$ denotes the (possibly empty) list of values specified for the variable X in $\mathbf{e}$.
- $\mathbf{1}(b)$ is the indicator function for the Boolean b, with $\mathbf{1}(b) = 1$ if b is true and 0 otherwise.

Equation (4.5) stipulates that the distribution for X' will follow the values assigned in the effect(s) provided that at least one effect specifies a value for it. If the effects include conflicting assignments, the distribution is spread uniformly over the alternative values. If no effects $e_1, ...e_n$ specifies a value for X , the value for X' is set to a default $None$ value with probability 1.

If the node X' is an update of an existing node X, the procedure remains essentially the same as for (4.5), except in the case where all the effects specify empty assignments for the variable. In such case, the distribution for X' will fall back to the value defined for the existing node X instead of being assigned a $None$ value:

$$P(X'=x' \mid r_1=e_1, ...r_n=e_n, X=x) = \begin{cases} \frac{\sum_{v \in \mathbf{e}(X)} \mathbf{1}(x'=v)}{|\mathbf{e}(X)|} & \text{if } \mathbf{e}(X) \neq \emptyset \\ \mathbf{1}(x'=x) & \text{otherwise} \end{cases} \tag{4.6}$$

As an example, the output distribution $P(A' \mid r_5 = \{\cdot\}, r_6 = \{A = a_2 \wedge E = e_2\}, A = a_3)$ in Figure 4.3 results in a deterministic distribution with a unique value a_2 with probability 1, since $\mathbf{e}(A) = \{a_2\}$. If the two rules generate conflicting assignments, the probability mass is divided equally over the alternative values. The output distribution $P(A' \mid r_5 = \{A = a_1\}, r_6 = \{A = a_2 \wedge E = e_2\}, A = a_3)$ provides two alternative values for A: $\mathbf{e}(A) = \{a_1, a_2\}$. The output distribution is thus a uniform distribution with two values: a_1 and a_2, each with probability 0.5. Finally, if all effects are empty, the output distribution is a simple copy of the distribution for the existing variable: $P(A' \mid r_5 = \{\cdot\}, r_6 = \{\cdot\}, A = a_3)$ has a unique value a_3 with probability 1.

Output distributions are directly derived from the effects in the rule nodes and are thus entirely parameter-free. The ordering of the parent rules in the conditional probability distribution is arbitrary. The resulting distribution bears resemblance to the probabilistic Independence of Causal Influence (pICI) described by Díez and Druzdzel (2006).

Instantiation algorithm

The procedure for instantiating a rule in a given dialogue state is detailed in Algorithm 4.

The first steps of the instantiation process are to extract in the Bayesian network the input variables of the rule (line 1), create a node corresponding to the rule (line 2) and include its dependency edges in the network (line 3). The algorithm then checks whether at least one effect in r is non-empty given its conditional dependences (lines 4). If all effects are empty, the rule node is irrelevant and can be directly pruned (line 5). Otherwise, the output variables are extracted (line 7), and output nodes that do not already exist in the network are created (line 10-11). The final step is to establish dependency edges between the rule node and these output variables (line 13).

4.3.2 Utility rules

Utility rules are instantiated in the Bayesian network according to a similar procedure, with two notable differences compared to probability rules:

Algorithm 4 : INSTANTIATEPROBRULE ($\mathcal{B}, r$)

Input: Bayesian network $\mathcal{B}$ for the current state
Input: Probability rule r to instantiate in network

1: $I_1, ...I_k \leftarrow$ input variables for r
2: Create chance node r with the rule distribution in Eq. (4.4)
3: Add node r and dependency edges $I_1, ...I_k \rightarrow r$ to $\mathcal{B}$
4: **if** $Val(r) = \{\cdot\}$ **then**
5: Prune r from $\mathcal{B}$
6: **else**
7: $O_1, ...O_l \leftarrow$ output variables mentioned in the effects of r
8: **for all** variable $O \in O_1, ...O_l$ **do**
9: **if** O' not already in $\mathcal{B}$ **then**
10: Create chance node O' with the output distribution in Eq. (4.5)-(4.6)
11: Add node O' and (in case O exists) dependency edge $O \rightarrow O'$ to $\mathcal{B}$
12: **end if**
13: Add dependency edge $r \rightarrow O'$ to $\mathcal{B}$
14: **end for**
15: **end if**

- As utility rules define utility distributions, their instantiation correspond to utility nodes instead of chance nodes.

- Instead of output variables, the rules are associated with decision variables. The association direction is inverted, as the decision node must be input to the utility node.

The result of the instantiation process is a decision network that incorporates chance nodes (corresponding to the state variables), utility nodes (corresponding to the utility rules) and associated decision nodes. Figure 4.4 illustrates the instantiation of two utility rules r_7 and r_8.

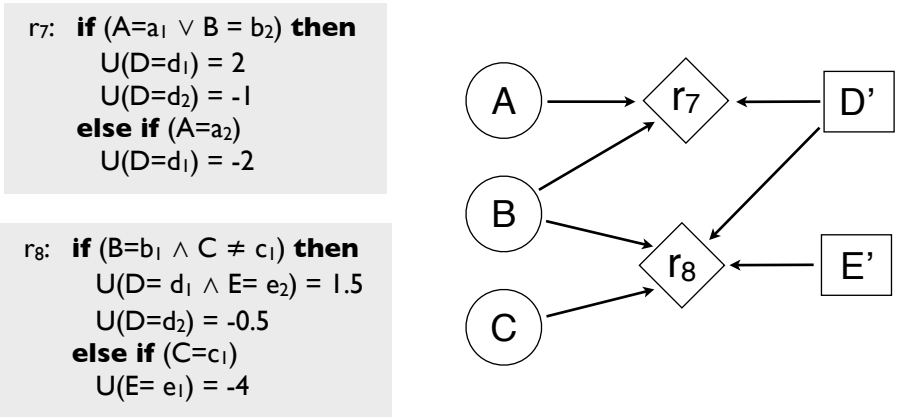


Figure 4.4: Example of instantiation for the two utility rules r_7 and r_8.

The utility distribution associated with each rule is a direct translation of the *if ... then ... else* rule. Formally, the utility distribution generated by a rule r with input variables $I_1, ...I_k$ and

decision variables $A_1, ...A_l$ is defined as:

$$U_r(I_1{=}i_1, ...I_k{=}i_k, A_1{=}a_1, ...A_l{=}a_l) = U_i(A_1{=}a_1 \wedge ...A_l{=}a_l) \quad (4.7)$$
$$\text{where } i = \min_i(c_i \text{ is satisfied with } I_1{=}i_1 \wedge ...I_k{=}i_k)$$

If no utility is explicitly specified for $A_1{=}a_1 \wedge ...A_l{=}a_l$, the default value is zero.

As is conventionally assumed in decision networks, the total utility for a given assignment of decision variables is defined as the *sum* of all utilities. In case $A{=}a_1$, $B == b_1$ and $C{=}c_1$, we can therefore calculate the total utility for the actions $D'{=}d_1 \ \wedge \ E'{=}e_1$ to be equal to $2 - 4 = -2$.

Instantiation algorithm

The procedure for instantiating a utility rule is similar in most respects to the one already outlined for probability rules. Algorithm 5 details the procedure, starting from the extraction of the input variables, the creation of the rule node, and the inclusion of conditional dependences (line 1-3). The algorithm then checks if the utility distribution stipulates a non-zero utility for at least one decision (line 4). If the answer is negative, the node is essentially irrelevant and can be pruned (line 5). The decision variables associated with the rule are extracted (line 7), and a corresponding decision node is created if it does not already exists (line 10). Finally, the possible values specified for the decision variable are integrated to the node (line 12), and a dependency edge is established between the decision node and the utility node for the rule (line 13).

Algorithm 5 : INSTANTIATEUTILRULE ($\mathcal{B}, r$)

Input: Bayesian network $\mathcal{B}$ for the current state
Input: Utility rule r to instantiate in network

1: $I_1, ...I_k \leftarrow$ input variables for $rule$
2: Create utility node r with the utility distribution in Eq. (5.7)
3: Add node r and dependency edges $I_1, ...I_k \rightarrow r$ to $\mathcal{B}$
4: **if** utility distribution is empty for all inputs **then**
5: Prune r from $\mathcal{B}$
6: **else**
7: $A_1, ...A_l \leftarrow$ decision variables mentioned in the effects of r
8: **for all** variable $A \in A_1, ...A_l$ **do**
9: **if** A' not already in $\mathcal{B}$ **then**
10: Create decision node A' and add it to $\mathcal{B}$
11: **end if**
12: Add in $Val(A')$ the action values specified in the effects of r
13: Add dependency edge $A' \rightarrow r$ to $\mathcal{B}$
14: **end for**
15: **end if**

4.3.3 Quantification

We saw in Section 4.2.3 that conditions and effect could include universally quantified variables, but have not yet discussed how such underspecified rules could be practically instantiated in the

Bayesian network. The general instantiation principle remains unchanged: to each rule corresponds a distinct rule node responsible for the mapping between input and output variables (or decision variables for utility rules). The instantiation procedure must be however extended to accommodate the presence of quantified variables. The key idea is to find all relevant *groundings* for the quantified variables, and then calculate the effect distribution for each grounding. This method of handling quantifiers by extracting all possible groundings and reasoning at the propositional level is an instance of *ground inference* (Getoor and Taskar, 2007).

Extraction of input variables

Universally quantified rules may underspecify both the names and values of random variables, as we saw in the examples of Section 4.2.3. Rule r_3 includes for instance a reference to an underspecified random variable $shape(y)$. In order to instantiate the rule, the system must therefore first determine all random variables included in the model that match the underspecified description. If rule r_3 is instantiated on a state containing two objects o_1 and o_2, the resulting input variables will therefore be $shape(o_1)$ and $shape(o_2)$.

Algorithm 6 outlines how this search for matching input variables can proceed. The first step is to extract the initial input variables associated with the rule, which may include underspecified descriptions such as $shape(y)$. The algorithm then loops on each underspecified description to find possible groundings in the random variables of the Bayesian network. The final result corresponds to the combination of the fully specified input variables for the rule and the groundings for the underspecified variables.

Algorithm 6 : GETINPUTVARIABLES ($\mathcal{B}, r$)

Input: Bayesian network $\mathcal{B}$ for the current state
Input: Probability or utility rule r

1: $\mathcal{I}_r \leftarrow$ Initial (possibly underspecified) input variables for r
2: $\mathcal{U}_r \leftarrow$ Subset of variable names in $\mathcal{I}_r$ that are underspecified
3: $\mathcal{G}_r \leftarrow$ Set of possible groundings for $\mathcal{U}_r$, initially empty
4: **for** underspecified variable name $u \in \mathcal{U}_r$ **do**
5: **for** random variable X in $\mathcal{B}$ **do**
6: **if** X matches u **then**
7: $\mathcal{G}_r \leftarrow \mathcal{G}_r \cup [X]$
8: **end if**
9: **end for**
10: **end for**
11: **return** $(\mathcal{I}_r \,/\, \mathcal{U}_r) \cup \mathcal{G}_r$

Line 1 in Algorithms 4 and 5 is then replaced by:

1: $I_1, ...I_k \leftarrow$ GETINPUTVARIABLES ($\mathcal{B}, r$)

Extraction of relevant groundings

Once the input variables for the rules are retrieved, the next step is to establish the set of relevant groundings G for the universally quantified variables. The groundings are always determined relative to a particular assignment of values for the (grounded) input variables $I_1, ...I_k$.

Given an input assignment $\{I_1 = i_1 \wedge ...I_k = i_k\}$, the set of groundings G are derived in a heuristic manner, by checking which ground terms (constants and functions of constants) from the input assignment can function as proper substitutions for the quantified variables. These ground terms define the domain of discourse for the application of the universal quantifier. The collection of relevant ground terms is then combined into subsets of size p, where p corresponds to the number of universally quantified variables $\mathbf{y} = y_1, ...y_p$. These combinations form the groundings G for the input assignment.

Quantified probability rules

As we have seen, probability rules are instantiated as chance nodes associated with a rule distribution. For rules containing universal quantifiers, each grounding in G gives rise to a particular distribution over effects. The instantiation procedure generates a distinct effect distribution for each grounding $\mathbf{g}_j$:

$$\begin{aligned} P(r_{\mathbf{g}_j} = e' \mid I_1 = i_1, ...I_k = i_k) &= P(E_i = e) \\ &\text{where } i = \min_i(c_i[\mathbf{y}/\mathbf{g}_j] \text{ is satisfied with } I_1 = i_1 \wedge ...I_k = i_k) \\ &\text{and } e' = e[\mathbf{y}/\mathbf{g}_j] \end{aligned} \tag{4.8}$$

The expression $\phi[a/b]$ denotes (as in formal logic) the formula ϕ where all instances of a are substituted by b.

Empty and redundant effect distributions are discarded. The grounding procedure results in a collection of distributions $\langle P(r_{\mathbf{g}_1}), ...P(r_{\mathbf{g}_p})\rangle$. The final conditional probability distribution for the rule node is then defined as the joint distribution over these grounding-specific distributions:

$$P(r = [e_1 \wedge ...e_q] \mid I_1 = i_1, ...I_k = i_k) = \prod_{j=1}^{q} P(r_{\mathbf{g}_j} = e_j) \tag{4.9}$$

Figure 4.5 shows how the rule r_3 is instantiated in a state with two objects o_1 and o_2, each associated with a random variable describing its shape.

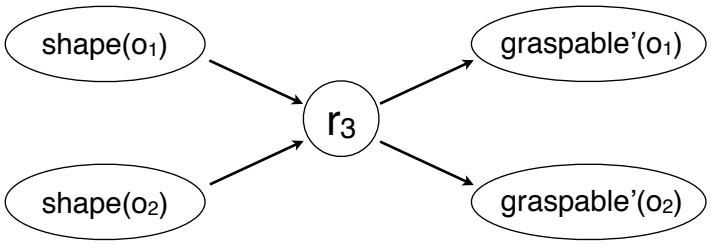


Figure 4.5: Instantiation of the probability rule r_3 on a state with two objects o_1 and o_2.

As an example, the probability distribution $P(r_3 \mid shape(o_1) = sphere, shape(o_2) = cone)$ has two relevant groundings $y = o_1$ and $y = o_2$, from which four alternative effects are derived:

- $\{graspable(o_1) = true \wedge graspable(o_2) = true\}$ with probability $0.9 \times 0.2 = 0.18$

- $\{graspable(o_1) = true \wedge graspable(o_2) = false\}$ with probability $0.9 \times 0.8 = 0.72$
- $\{graspable(o_1) = false \wedge graspable(o_2) = true\}$ with probability $0.1 \times 0.2 = 0.02$
- $\{graspable(o_1) = false \wedge graspable(o_2) = false\}$ with probability $0.1 \times 0.8 = 0.08$.

Quantified utility rules

Utility rules can be similarly extended to accommodate quantified variables. As for probability rules, the instantiation of universally quantified utility rules proceeds by determining a set of groundings and generating a particular utility distribution for each.[6] For a utility rule with input variables $I_1, ...I_k$, decision variables $A_1, ...A_l$ and quantified variables $\mathbf{y}$,

$$\begin{aligned} U_{\mathbf{g}_j}(I_1 = i_1, ...I_k = i_k, A_1 = a'_1, ...A_l = a'_l) = U_i(A_1 = a_1 \wedge ...A_l = a_l) \\ \text{where } i = \min_i(c_i[\mathbf{y}/\mathbf{g}_j] \text{ is satisfied with } I_1 = i_1 \wedge ...I_k = i_k) \\ \text{and } a'_1 = a_1[\mathbf{y}/\mathbf{g}_j], ... \ a'_l = a_l[\mathbf{y}/\mathbf{g}_j] \end{aligned} \tag{4.10}$$

After discarding empty and redundant distributions, the result is a set of utility distributions $\langle U_{\mathbf{g}_1}, ...U_{\mathbf{g}_p} \rangle$. The total utility distribution for the rule is then constructed by adding up the grounding-specific utility distributions:

$$U_r(I_1, ...I_k, A_1 ...A_l) = \sum_{j=1}^{q} U_{\mathbf{g}_j}(I_1, ...I_k, A_1, ...A_l) \tag{4.11}$$

The result of instantiating rule r_4 on a state with two objects o_1, o_2 with associated random variables $graspable(o_1)$ and $graspable(o_2)$ is shown in Figure 4.6.

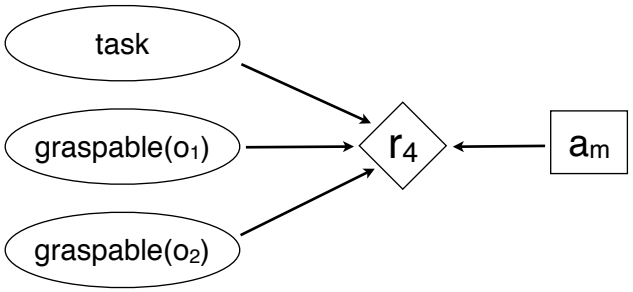


Figure 4.6: Instantiation of the quantified utility rule r_9 on a state with two objects o_1 and o_2.

The utility distribution $U_{r_4}(task = grasp(o_1), graspable(o_1) = true, graspable(o_2) = false, a_m)$ assigns the action $a_m = grasp(o_1)$ to a utility of 2, while the action $a_m = grasp(o_2)$ is assigned to a utility of -2.

[6] The extraction of groundings is slightly modified for utility rules in order to integrate the ground terms appearing in both the input and decision assignments.

Tractability aspects

Although the use of universally quantifiers greatly improves the expressivity of probabilistic rules, they also tend to increase the in- and out-degrees of rule nodes (that is, the cardinality of their parents and children nodes). Approximate inference techniques are thus necessary to handle this conditional structure in a tractable manner. Sampling methods such as likelihood weighting have in practice proved to work well in this setting (cf. Section 3.1.2).

It should be stressed that the groundings are always extracted *given a specific value assignment* for the input variables. By restricting the groundings to this limited domain of discourse, we ensure that the number of alternative effects remains bounded and avoid the generation of spurious effects. This instantiation procedure was found to be much more efficient than copying the rule in distinct nodes, as investigated in earlier implementations of the formalism (Lison, 2012c).

4.4 Processing workflow

The two previous sections detailed how probability and utility rules are internally defined, and how they can be instantiated as nodes of a graphical model. We are now ready to explain how collections of rules are practically applied at runtime to update the dialogue state and perform action selection. The general workflow is strongly inspired by information-state approaches to dialogue management (Larsson and Traum, 2000b; Buckley and Benzmüller, 2007), as the dialogue state serves as a central blackboard monitored by various groups of rules that are "triggered" upon relevant changes.

4.4.1 Domain representation

Dialogue domains can consist of multiple probability and utility rules. These rules are internally grouped in collections of rules called *models*. A model is simply a collection of rules that is associated with one or more "trigger" variables that specify when the model is to be instantiated. Each model is attached the dialogue state and monitors it to detect changes affecting their trigger variables. When these trigger variables are changed at runtime by another module, they lead to the instantiation of the model rules. Formally, a model m is defined as a pair $\langle \mathcal{T}_m, \mathcal{R}_m \rangle$ where $\mathcal{T}_m$ corresponds to the trigger variables and $\mathcal{R}_m$ to the rules included in the model.

A dialogue domain is represented as a pair $\langle \mathcal{B}_0, \mathcal{M} \rangle$, where $\mathcal{B}_0$ is the initial dialogue state and $\mathcal{M}$ the set of rule-based models attached to it. The organisation of rules into models allows the system designer to structure the application pipeline in a modular manner. Each model can be intuitively viewed as a distinct component responsible for a particular inference or decision step.

Section 7.2 explains how dialogue domains are practically encoded in the `openDial` architecture.

4.4.2 Update algorithm

The software architecture adopted in this thesis takes the form of an event-driven, blackboard architecture (Turunen, 2004; Buckley and Benzmüller, 2007) revolving around a dialogue state $\mathcal{B}$ represented as a Bayesian network. As in information state approaches, this dialogue state is read and written by the various modules integrated in the dialogue system.

The update procedure is shown in Algorithm 7. The procedure is started upon the reception of new variables to incorporate in the state, such as new user inputs processed by the ASR/NLU

components. The first step is to insert the variables in the dialogue state and possibly relate them to their predicted values (lines 2-3). The algorithm then triggers the instantiation of the relevant domain models (line 4), leading to a recursive chain of updates. If the expanded dialogue state contains decision and utility variables, the algorithms searches for the optimal action, selects it, and activates the models that are triggered as a result (lines 6-8). Finally, the updated state is reduced by pruning away unnecessary nodes and incorporating the evidence (line 10).

Algorithm 7 : UPDATESTATE ($\mathcal{B}, \mathbf{X}$)

Input: Bayesian network $\mathcal{B}$ for the current state
Input: New random variables $\mathbf{X}$ to insert in the state

1: Initialise evidence $\mathbf{e} \leftarrow \emptyset$
2: Insert $\mathbf{X}'$ to the current state $\mathcal{B}$
3: $\mathcal{B}, \mathbf{e} \leftarrow$ INTEGRATEPREDICTIONS($\mathcal{B}, \mathbf{e}, \mathbf{X}'$)
4: $\mathcal{B}, \mathbf{e} \leftarrow$ TRIGGERMODELS ($\mathcal{B}, \mathbf{e}, \mathbf{X}'$)
5: **while** $\mathcal{B}$ contains decision variables **do**
6: $\quad \mathbf{a}^* \leftarrow$ SELECTACTION ($\mathcal{B}, \mathbf{e}$)
7: $\quad$ Assign $\mathbf{A}' = \mathbf{a}^*$
8: $\quad \mathcal{B}, \mathbf{e} \leftarrow$ TRIGGERMODELS ($\mathcal{B}, \mathbf{e}, \mathbf{A}'$)
9: **end while**
10: $\mathcal{B} \leftarrow$ PRUNESTATE($\mathcal{B}, \mathbf{e}$)

We now describe each of these steps in detail.

Connecting predictions and observations

Rules are sometimes used to provide predictions on variables that will be observed in the next time steps.[7] In order to distinguish random variables that express a prediction on a future outcome from those that reflect an actual (although possibly uncertain) observation, we denote predictive variables with a subscript p. A variable X_p is thus a prediction for the future observation of the variable X.

Prediction and observation variables must be connected with one another at runtime. In the case where the observation is known with certainty, this connection can simply be represented as an assignment of evidence values. However, dialogue often include observations that are themselves uncertain and represent "soft" or virtual evidence. Several techniques are available to practically encode this evidence. The method adopted in this thesis is to add a new boolean-valued chance node, subsequently called the *equivalence node* eq_X, that is conditionally dependent on both X and X_p, as shown in Figure 4.7. The conditional probability distribution of eq_X is determinis-

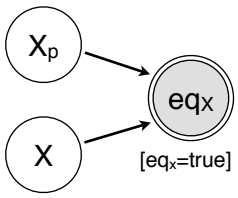


Figure 4.7: Equivalence node eq_X with parents X and X_p.

[7] This is notably the case for the user action model $P(a_u \mid i_u, a_m)$, which estimate the relative probabilities for the next dialogue action from the user. The prediction provide a prior on the future observation of the user action.

tic (graphically depicted by a double circle around the chance node):

$$P(eq_X = true \mid X = x, X_p = x_p) = \begin{cases} 1 & \text{if } x = x_p \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

The use of a distinct node to express the evidence is motivated by the fact that X and X_p can have arbitrary incoming and outgoing edges with other variables.

The assignment $eq_X = true$ is added to the evidence. The posterior distribution given the evidence allows the prediction to act as a prior for the observed distribution:

$$\begin{aligned} & P(X = x \mid eq_X = true) \\ & = \alpha\ P(X = x) \sum_{x_p \in Val(X_p)} P(eq_X = true \mid X = x, X_p = x_p) P(X_p = x_p) && (4.13) \\ & = \alpha\ P(X = x)\ P(X_p = x) && (4.14) \end{aligned}$$

The inclusion of an equivalence node between X and X_p with evidence $[eq_X = true]$ modifies the distribution of the variables X andX_p as well as their respective parents/children nodes. Algorithm 8 illustrates the process of integrating predictions for the variables $Vars$.

Algorithm 8 : IntegratePredictions ($\mathcal{B}$, $\mathbf{e}$, $Vars$)

1: **for all** $X \in Vars$ **do**
2: **if** there is a corresponding prediction variable $X_p \in \mathcal{B}$ **then**
3: Create equivalence node eq_X with distribution in Eq. (4.12)
4: Insert eq_X in $\mathcal{B}$ with parents X and X_p
5: Add assignment $[eq_X = true]$ to evidence $\mathbf{e}$
6: **end if**
7: **end for**
8: **return** $\mathcal{B}, \mathbf{e}$

Model instantiation

After inserting the new variables in the dialogue state and connecting them to their predicted values, the next step in the processing workflow is to trigger the relevant domain models .

Algorithm 9 summarises the steps involved in the instantiation of the domain models. The algorithm takes three arguments: a dialogue state $\mathcal{B}$ represented as a Bayesian network, an assignment of evidence values and a list of random variables that have been recently updated in the dialogue state. The algorithm loops on all domain models and instantiates the ones that are triggered by the updated variables. The rules are instantiated one by one, following the procedure we have outlined in the previous section. Once all models are traversed, the output variables of the instantiated rules become updated variables themselves, and the procedure is repeated until no more models can be applied. To avoid the occurrence of infinite triggering cycles, models are limited to one instantiation per update. The algorithm returns both the dialogue state expanded with new variables, and the evidence assignment attached to the equivalence nodes.

Algorithm 9 : TRIGGERMODELS ($\mathcal{B}, \mathbf{e}$, $UpdatedVars$)

while $UpdatedVars \neq \emptyset$ **do**
  $NewVars \leftarrow \emptyset$
  for all models m **do**
    if $UpdatedVars \cap \mathcal{T}_m \neq \emptyset$ and m has not yet been applied **then**
      for all rule $r \in \mathcal{R}_m$ **do**
        if r is a probability rule **then**
          $\mathcal{B} \leftarrow$ INSTANTIATEPROBRULE($\mathcal{B}, r$)
        else if r is a utility rule **then**
          $\mathcal{B} \leftarrow$ INSTANTIATEUTILRULE($\mathcal{B}, r$)
        end if
        Let $\mathcal{O}_r$ be the new output variables created by rule r
        $NewVars \leftarrow NewVars \cup \mathcal{O}_r$
        $\mathcal{B}, \mathbf{e} \leftarrow$ INTEGRATEPREDICTIONS($\mathcal{B}, \mathbf{e}, \mathcal{O}_r$)
      end for
    end if
  end for
  $UpdatedVars \leftarrow NewVars$
end while
return $\mathcal{B}, \mathbf{e}$

Action selection

Whenever the new dialogue state contains utility and decision nodes, the system must decide on the action to perform. Algorithm 10 illustrates how actions can be selected on the basis of the current dialogue state augmented with the decision and utility nodes created by the utility rules. The algorithm searches for the assignment of action values that maximise the current utility given the dialogue state and the evidence and returns it. This utility maximisation is based on standard inference algorithms for decision networks such as likelihood weighting (cf. Section 3.1.2).

The utility nodes are removed from the state once the decision is made. The action selection procedure described here only takes into account the current (immediate) utility and does not rely on any forward planning. Chapter 6 demonstrates how this procedure can be extended to perform online planning on a limited horizon.

Algorithm 10 : SELECTACTION ($\mathcal{B}, \mathbf{e}$)

Let $\mathbf{A}'$ be the set of all decision variables in $\mathcal{B}$
Find optimal value $\mathbf{a}^* = \text{argmax}_{\mathbf{a}} U(\mathbf{A}' = \mathbf{a}, \mathbf{e})$
Remove utility nodes from the state $\mathcal{B}$
return $\mathbf{a}^*$

State pruning

The instantiation of the domain models results in the integration of numerous new nodes in the dialogue state. However, many nodes in this expanded Bayesian network only serve as intermediaries and do not directly express meaningful information about the current state of the dialogue. The

last step is therefore to reduce the dialogue state to its minimal size, by removing all intermediary nodes – including rule nodes, outdated versions of state variables, equivalence nodes and predictive nodes that are attached to them – in order to only retain current state variables. The accumulated evidence is also integrated in the posterior distribution of the state variables.

The procedure is outlined in Algorithm 11. The first step is to determine which nodes to keep (line 1-6). Only the most recent versions of state variables are retained. The nodes are then added one by one in a new dialogue state $\mathcal{B}'$. The parents of each variable is determined, and its conditional probability distribution is calculated given the evidence. The parents of a state variable are the closest ancestors of the variable within the subset of nodes in $NodesToKeep$, and its conditional probability distribution is determined as $P_{\mathcal{B}}(N \mid Parents, \mathbf{e})$. This posterior distribution is calculated via sampling techniques. This is done by sampling all nodes in $\mathcal{B}$, then deriving the distributions $P_{\mathcal{B}}(N \mid Parents, \mathbf{e})$ on the basis of the collected samples.

Algorithm 11 : PRUNESTATE ($\mathcal{B}, \mathbf{e}$)

1: $NodesToKeep \leftarrow \emptyset$
2: **for all** node $N \in \mathcal{B}$ **do**
3: **if** N is a state variable and $\nexists\, N' \in \mathcal{B}$ **then**
4: $NodesToKeep \leftarrow NodesToKeep \cup [N]$
5: **end if**
6: **end for**
7: Create new state $\mathcal{B}' \leftarrow \emptyset$
8: **for all** node $N \in NodesToKeep$ **do**
9: Add node N to $\mathcal{B}'$ (with primes removed from node name)
10: $Parents \leftarrow \{M \in NodesToKeep : M$ is an ancestor of N and there is a path $M \rightarrow^{+} N$ without node in $NodesToKeep\}$
11: Add dependency edges between $Parents$ and N in $\mathcal{B}'$
12: Assign distributions $P_{\mathcal{B}'}(N \mid Parents) \leftarrow P_{\mathcal{B}}(N \mid Parents, \mathbf{e})$
13: **end for**
14: **return** $\mathcal{B}'$

Figure 4.8 illustrates the input and output of the pruning process. Note that the primes attached to the labels of output variables are deleted from the random variable names.

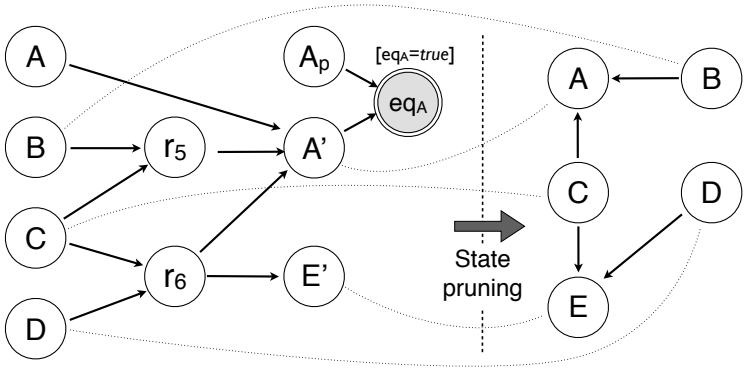


Figure 4.8: Illustration of the state pruning process. Only the nodes A', B, C, D and E' are retained. The dotted lines denote the correspondence between nodes.

4.4.3 Detailed example

We now describe a minimal but complete example of workflow for a short interaction.

Description

Assume a domain similar to the one shown in Figure 2.3, where a user can request a robot to move forward, backward, left, right, or stop. The set of dialogue acts a_u that can be recognised by the system is the following:

$$\{Request(Forward),\ Request(Backward),\ Request(Left),\\ Request(Right),\ Request(Stop),\ Other\}.$$

The corresponding system actions a_m are:

$$\{Move(Forward),\ Move(Backward),\ Move(Left),\\ Move(Right),\ Move(Stop),\ AskRepeat\}.$$

The objective of the system is to fulfil the user command if it is reasonably confident regarding which action to execute. Otherwise, the system asks the user to repeat.

Domain specification

The domain specification designed for this constructed example is constituted of an empty initial state and the following two rule-based models:

- Model m_1 is triggered by a_u and includes the two utility rules r_9 and r_{10}:

$$\begin{aligned} r_9 :\ & \forall y : \\ & \textbf{if}\ (a_u = Request(y))\ \textbf{then} \\ & \quad \Big\{ U(a_m = Move(y)) = 2 \\ & \textbf{else} \\ & \quad \Big\{ U(a_m = Move(y)) = -2 \\ r_{10} :\ & \quad \Big\{ U(a_m = AskRepeat) = 0.5 \end{aligned}$$

  Rule r_9 specifies that the utility of executing the action corresponding to the user command is 2, with a penalty of -2 when the wrong action is executed. Rule r_{10} assign a utility of 0.5 for asking a clarification question.[8]

- Model m_2 is triggered by a_m and has one single predictive rule r_{11}:

$$\begin{aligned} r_{11} :\ & \forall y : \\ & \textbf{if}\ (a_m = AskRepeat \wedge a_u = y)\ \textbf{then} \\ & \quad \Big\{ P(a_{u\text{-}p} = y) = 0.9 \end{aligned}$$

[8] As the action selection process presented thus far does not perform forward planning, the utilities provided in this example correspond to long-term expected utilities (Q-values in the reinforcement learning terminology).

Rule r_{11} specifies that the probability that the user will repeat his last utterance when asked by the system to do so is expected to be 0.9.

Processing workflow

We now detail the processing workflow associated with the following constructed interaction:

USER : Now move forward
$\tilde{a}_u = \langle (Request(Forward), 0.6), (Request(Backward)), 0.4) \rangle$

SYSTEM : Could you please repeat?

USER : Please move forward!
$\tilde{a}_u = \langle (Request(Forward), 0.7), (Other, 0.2), (Request(Backward), 0.1) \rangle$

SYSTEM : OK, moving forward!

The (constructed) recognition hypotheses $\tilde{a}_u$ produced by the ASR/NLU components are written underneath the user utterance.

Figure 4.9 details the steps involved in the state update procedure that follows from the reception of dialogue act hypotheses from the natural language understanding component.

Step 1 inserts the new dialogue act hypotheses on the dialogue state. This insertion triggers the utility model m_1. The instantiation results in Step 2 in the creation of two utility nodes and one decision node. The optimal action to perform in such case is $AskRepeat$, which is selected by the system in Step 3. The action selection triggers model m_2 in Step 4, which creates a prediction node $a'_{u\text{-}p}$ expressing the expected probability distribution for the next user dialogue act. The state is finally pruned of the intermediary rule node in Step 5. System components such as NLG can react on the updated state and generate the proper linguistic realisation of the system action. The system then waits for the user input, which is shown in Step 6. The relation between the predicted and actual user response leads in Step 7 to the creation of an equivalence node, and the inclusion of the assignment $eq_{a_u} = true$ in the evidence. We notice that the combination of the prior distribution over predicted values and the actual distribution over dialogue act hypotheses increases the probability of $a'_u = Request(Forward)$. Step 8 triggers the model m_1 based on the new user input. The optimal action is this case is $Move(Forward)$, which is selected in Step 9. This selection triggers model m_2, but rule r_{11} only generates in this case an empty effect and is therefore directly deleted. Finally, the state is pruned of its intermediary nodes in Step 10, retaining only the last user and system actions a_u and a_m.

In comparison to the finite-state solution present in Figure 2.3, we observe that the rule-structured approach defined by models m_1 and m_2 allows the dialogue manager to accumulate evidence over time and prime the recognition hypotheses of the user dialogue act a_u based on the previous dialogue act. This accumulation of evidence is absent from the FSA, due its rigid state representation and lack of memory.

4.5 Advanced modelling

Dialogue domains often include random variables with values expressed via specific data structures such as lists or strings. The rule-based formalism described in the previous sections can be easily

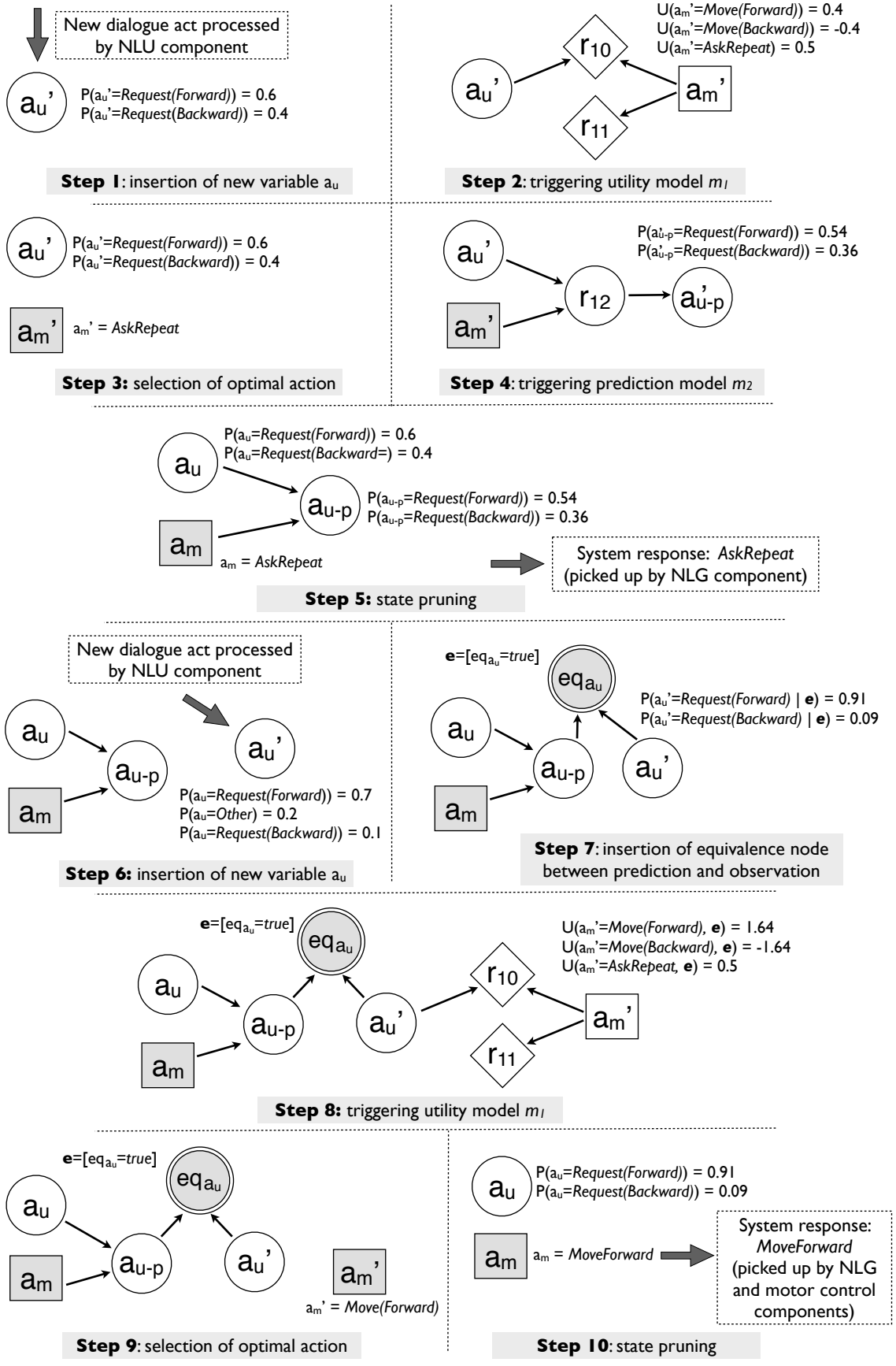


Figure 4.9: Detailed example of processing workflow.

complemented with special-purpose tools to efficiently operate on these data structures. We first explain how conditions and effects can be defined on variables that represent lists, and then discuss how rules can manipulate strings.

4.5.1 Operations on lists

Some state variables are best represented as lists of elements. For instance, the dialogue state may include random variables that enumerate the n most recent dialogue acts in the interaction history, the stack of tasks that remain to perform, or the list of visual objects perceived by the system. The range of values for such state variables is the power set of its possible elements.

Special-purpose operators for the manipulation of such lists can be integrated in both the conditions and effects of probabilistic rules:

- Rule conditions can include operators to check the presence or absence of particular elements in a list, such as $a \in A$ or $a \notin A$.
- Rule effects can also be augmented to manipulate elements from a list. Three new types of effects are created to this end, in addition to the traditional assignment of output values: *add effects* (adding an element to a list), *delete effects* (deleting an element from a list) and *clear effects* (clearing all elements of a list). The resulting lists are sorted by insertion order.

Figure 4.10 illustrates two rules that apply these new effects to update a state variable A.

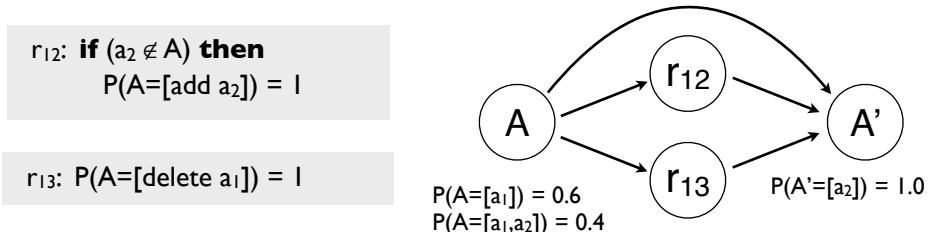


Figure 4.10: Example of rules using add/delete effects to manipulate lists.

These new effects can be incorporated to the framework through a simple modification of the output distribution. Let $\mathbf{e}$ denote as before the conjunction of all effects $e_1 \wedge ...e_n$. In addition to the previously defined set of values $\mathbf{e}(X)$ assigned for the variable X, we construct two new sets of values $\mathbf{e}_{add}(X)$ and $\mathbf{e}_{del}(X)$ that represent the values that are respectively added and deleted for the variable X through the new effects we just described. Note that $\mathbf{e}_{del}(X)$ may include all values for X if the clear effect is applied.

The output distribution in Equation (4.6) is then rewritten as:

$$P(X'=x' \mid r_1=e_1, ...r_n=e_n, X=x) = \begin{cases} \dfrac{\sum_{v \in \mathbf{e}(X)} \mathbf{1}(x'=v)}{|\mathbf{e}(X)|} & \text{if } \mathbf{e}(X) \neq \emptyset \\ \mathbf{1}(x' = (\mathbf{e}_{add}(X) \cup (x \;/\; \mathbf{e}_{del}(X)))) & \text{otherwise} \end{cases} \tag{4.15}$$

The output distribution associated for a new variable (cf. Equation (4.5)) can be rewritten in a similar manner.

4.5.2 Operations on strings

Many of the data structures present in the dialogue state are strings – the most prominent ones being the last user utterance u_u and the last system utterance u_m. The integration of special-purpose functionalities for manipulating strings within the conditions and effects of probabilistic rules is therefore desirable. In particular, rules can be extended to perform template-based string matching operations. The idea is to include a new type of conditions that checks whether a string matches a given template. Both full and partial matching can be employed. Templates are allowed to include slots to fill. These slots are conceptually similar to the quantified variables discussed in Sections 4.2.3 and 4.3.3. A successful match will thus generate values for the filled slots, which will be included as part of the groundings for the rule.

Figure 4.11 illustrate how such rules are applied in practice. $\{OBJ\}$ denotes a slot that is to be filled through matching the template with the value specified in u_u.

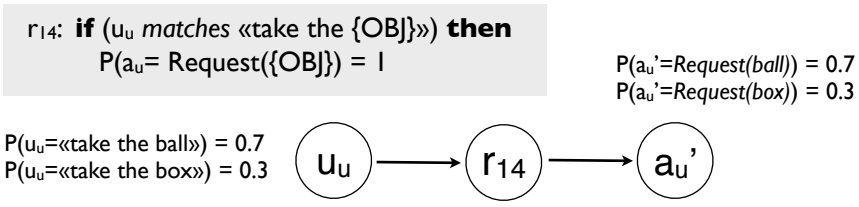


Figure 4.11: Example of rule using string matching operations.

4.6 Relation to previous work

The idea of using structural knowledge in probabilistic models has been explored in many directions, both in the fields of decision-theoretic planning and reinforcement learning (Hauskrecht et al., 1998; Pineau, 2004a; Kersting and Raedt, 2004; Lang and Toussaint, 2010; van Otterlo, 2012) and in statistical relational learning (Jaeger, 2001; Richardson and Domingos, 2006; Getoor and Taskar, 2007). The introduced structure may be hierarchical, relational, or both. As in our approach, most of these frameworks rely on the use of expressive representations serving as templates for the generation of classical probabilistic models. The surveys of van Otterlo (2006, 2012) provide a complete overview of relational and first-order logical approaches for reinforcement learning in Markov decision processes, covering both model-free and model-based methods. While the formalisation presented in this thesis and the aforementioned approaches share many insights, they also reveal several interesting differences:

- Probabilistic rules are primarily tailored for dialogue management tasks and seek to capture dialogue domains by striking a balance between propositional and first-order logic. The formalism deliberately eschews the complexity of full-scale first-order probabilistic inference to ensure that the domains models can be applied under real-time constraints. This design choice sets it apart from other frameworks such as Markov Logic Networks which can express arbitrary first-order formulae but are often tedious to instantiate due to the size and

complexity of the resulting models.[9]

- Probabilistic rules are also designed to operate under partially observable settings, as state uncertainty is a pervasive and unavoidable aspect of verbal interactions. By contrast, most previous work on relational probabilistic models are limited to fully observable environments, with the exception of some limited theoretical studies by Wang and Khardon (2010); Sanner and Kersting (2010).

- Finally, the presented framework posits that the *if ... then ... else* structures of probabilistic rules are best encoded by the system designers based on their expert knowledge of the domain, while the rule parameters can be estimated empirically. We therefore exclude the problem of structure learning from the scope of this thesis, as opposed to several approaches in which the domain rules and constraints are extracted via machine learning techniques (Pasula et al., 2007; Kok and Domingos, 2009).

Probabilistic rules also bear similarities with planning description languages such as the Planning Domain Description Language (PDDL, see McDermott et al., 1998) and its probabilistic extension, the Probabilistic Problem Description Language (PPDDL, see Younes and Littman, 2004). These languages are structured through action schemas that specify how (parametrised) actions can yield particular effects under various conditions. As in probabilistic rules, these languages try to carefully balance between the language expressivity and the complexity of the planning algorithm, based on a subset of first-order logic. A relational extension of PDDL, named RDDL, has also been introduced in recent planning competitions (Sanner, 2010). The learning techniques presented by Pasula et al. (2007) to estimate transition functions based on noisy indeterministic deictic rules is directly related to our approach, as is the recent work Lang and Toussaint (2010) on probabilistic noisy planning rules. Both frameworks define conditions associated with probabilistic distributions over effects. Their approaches are however restricted to fully observable settings.

In the dialogue management literature, most structural approaches rely on a clear-cut task decomposition into goals and sub-goals (Allen et al., 2000; Steedman and Petrick, 2007; Bohus and Rudnicky, 2009b), where the completion of each goal is assumed to be fully observable, discarding any remaining uncertainty. Our own work on multi-policy dialogue management in Lison (2011) relaxes the assumption of perfect knowledge of task completion, handling multiple policies as a problem of probabilistic inference over activation variables. Probabilistic rules can be considered an extension of this early work, where the structural knowledge is not confined to task decomposition but is extended to generic rules over state variables.

The formalism presented in this chapter is strongly inspired by information-state approaches to dialogue management (Larsson and Traum, 2000a; Bos et al., 2003), which are also based on a shared state representation that is updated according to a rich repository of rules. Ginzburg (2012) also models conversational phenomena by way of update operations that are encoded with rules mapping conditions to effects. However, contrary to the framework presented here, the rules specified in these approaches are generally deterministic and do not include learnable parameters. The action selection mechanism is also conceptualised slightly differently, as information-state frameworks rely on rules that directly select the most appropriate action given the current state. Prob-

[9] See however Kennington and Schlangen (2012) for an approach that attempts to apply Markov Logic Networks to natural language understanding tasks.

abilistic rules adopt by contrast a decision-theoretic approach that divides action selection in two stages: rules first provide utility distributions for the system action, and the system then searches for the action that yields the maximum expected utility.

The literature on dialogue policy optimisation with reinforcement learning also contains several approaches dedicated to dimensionality reduction for large state-action spaces, such as function approximation (Henderson et al., 2008), hierarchical reinforcement learning (Cuayáhuitl et al., 2010) and summary POMDPs (Young et al., 2010). Many of these techniques have already been discussed in Section 3.3 and will therefore not be repeated here. Most current approaches in dialogue policy optimisation focus on large but weakly structured state spaces (generally encoded as large lists of features), which are suited for slot-filling dialogue applications but are difficult to transfer to more open-ended or relational domains. The idea of state space partitioning, implemented here via high-level conditions, has also been explored in recent papers (see e.g. Williams, 2010). Crook and Lemon (2010) explored the introduction of complex user goal states including disjunction and negation operators. Cuayáhuitl (2011) describes a policy optimisation approach based on logic-based representations of the state-action space for relational MDPs. The main difference with our approach lies in his reduction of the belief state to fully observable variables whereas we retain the partial observability associated with each variable. The work of Mehta et al. (2010); Raux and Ma (2011) demonstrated how tree-structured Bayesian networks called probabilistic ontology trees can improve belief tracking performance. The tree structure is derived in their work from a hierarchical concept structure . Finally, O'Neill et al. (2011) describe a procedure for dialogue strategy selection based on probabilistic logic.

4.7 Conclusion

This chapter presented the formalism of probabilistic rules, which forms the core of the modelling approach developed in this thesis. We started by arguing that dialogue models are often highly structured, and that this structure can be leveraged by (1) introducing latent variables, (2) partitioning value assignments for the parent variables, and (3) making use of quantification. We then explained how these structural insights can be transferred into a new framework – probabilistic rules – that combines concepts borrowed from both first-order logic and probability theory in order to get "the best of both worlds", i.e. a representation formalism that is both richly expressive and capable of capturing uncertain knowledge. These rules are practically defined as *if...then...else* control structures that associate high-level conditions on input variables to probabilistic effects on output variables. Multiple extensions of the formalism have been developed to e.g. encode utility distributions, enclose universal quantifiers, and efficiently manipulate data structures such as lists and templates.

At runtime, these rules are instantiated in the Bayesian network representing the current dialogue state. The instantiation procedure creates a latent node for each rule, which is conditionally dependent on the input variables of the rule. For probability rules, this node is a chance node that expresses a probability distribution over the possible effects of the rule. Utility rules are similarly instantiated with a utility node expressing the utility distribution for specific decision variables. Universally quantified variables can be included in the conditions and effects of the rules, allowing particular aspects of the rule to be underspecified. The rules are grouped into models that are attached to the dialogue state and are triggered upon relevant state updates.

Probability and utility rules effectively function as high-level templates for the definition of a dynamic decision network. The expressive power of these rules allows them to efficiently encode complex relations between variables, and thereby reduce the number of parameters to estimate. We have however not yet detailed how this parameter estimation is practically performed. The next two chapters provide answers to this important question.

Chapter 5

Learning from Wizard-of-Oz data

The previous chapter outlined the formalism of probabilistic rules and their instantiation in the dialogue state. Probabilistic rules are generally associated to a number of parameters, which may either express probabilities over effects (for probability rules) or utilities associated with particular decisions (for utility rules). But where do these probabilities and utilities exactly come from?

We have developed in this thesis two distinct approaches to this question. The present chapter concentrates on the first approach, which is grounded in supervised learning techniques. We demonstrate how rule parameters can be optimised to best imitate the decision choices of a human expert through a process of statistical estimation based on Wizard-of-Oz data. The next chapter will then detail an alternative, reinforcement learning approach to the same task.

The chapter is divided in three sections. Section 5.1 describes how uncertainty regarding the value of rule parameters can be explicitly represented through prior distributions. Section 5.2 spells out how these distributions can be gradually refined through Bayesian learning on data gathered from Wizard-of-Oz interactions. The learning algorithm is used to progressively narrow down the spread of the parameter distributions to the values providing the best fit for the training data. Finally, Section 5.3 presents experimental results in a human-robot interaction domain for which small amounts of Wizard-of-Oz data were recorded. The experiment compared the learning performance of a utility model structured with probabilistic rules against two baselines respectively encoded with plain utility tables and with linear models. The evaluation showed that the rule-structured model was able to imitate the dialogue policy followed by the wizard significantly better than its unstructured counterparts.

5.1 Parameters of probabilistic rules

5.1.1 Generalities

The probabilistic rules described so far all relied on fixed probability and utility values. These values can however be replaced by parameters that reflect unknown values that are to be estimated empirically, on the basis of training data.

The overall structure of parametrised rules remains essentially identical to the one outlined in the previous chapter. Parametrised probability rules are once more defined in terms of conditions c_i associated to probability distributions $P(E_i)$ over effects. The probability of each effect $e_{(i,j)}$ is however no longer fixed but is instead represented by a parameter $\theta_{(i,j)}$, giving rise to the following

rule skeleton:

$$\begin{aligned}
&\textbf{if } (c_1) \textbf{ then} \\
&\quad \begin{cases} P(E_1 = e_{(1,1)}) = \theta_{(1,1)} \\ \ldots \\ P(E_1 = e_{(1,m_1)}) = \theta_{(1,m_1)} \end{cases} \\
&\ldots \\
&\textbf{else} \\
&\quad \begin{cases} P(E_n = e_{(n,1)}) = \theta_{(n,1)} \\ \ldots \\ P(E_n = e_{(n,m_n)}) = \theta_{(n,m_n)} \end{cases}
\end{aligned} \tag{5.1}$$

As the θ values represent probability values, they must satisfy as before the two probability axioms $\theta_{(i,j)} \geq 0 \; \forall i, j$ and $\sum_{j=1}^{m_i} \theta_{(i,j)} = 1 \; \forall i$.

Parametrised utility rules are analogously expressed by replacing fixed utility values with unknown parameters:

$$\begin{aligned}
&\textbf{if } (c_1) \textbf{ then} \\
&\quad \begin{cases} U_1(d_{(1,1)}) = \theta_{(1,1)} \\ \ldots \\ U_1(d_{(1,m_1)}) = \theta_{(1,m_1)} \end{cases} \\
&\ldots \\
&\textbf{else} \\
&\quad \begin{cases} U_n(d_n^1) = \theta_{(n,1)} \\ \ldots \\ U_n(d_{(n,m_n)}) = \theta_{(n,m_n)} \end{cases}
\end{aligned} \tag{5.2}$$

We shall focus in this work on the problem of parameter estimation given a known rule structure. The methodological stance adopted in this thesis is that the structure of probabilistic rules is best defined by the system designer, while the rule parameters is best determined by statistical optimisation techniques. Based on our own practical experience with various dialogue systems, we believe that such division of labour between the human designer and the learning algorithm is a sensible one, as system designers generally have a good grasp of the domain structure and relations between variables, but are often unable to quantify the precise probability of an effect or utility of an action.[1]

It should nevertheless be noted that some approaches have been recently developed in the literature on statistical relational learning (see e.g. Pasula et al., 2007; Kok and Domingos, 2009) to automatically extract both the structure and parameters of stochastic rules from raw data. These

[1] Humans are indeed notoriously poor at estimating probabilities and are prone to multiple cognitive biases, as evidenced by numerous studies in behavioural psychology. The interested reader is invited to consult e.g. Kahneman et al. (1981); Morgan and Henrion (1992) for more details on the psychological aspects of the human perception of uncertainty and the difficult problem of probability elicitation from experts.

methods are however generally confined to domains of limited size and full observability and are therefore difficult to apply to the types of domains investigated in this thesis.

5.1.2 Parameter priors

We follow in this thesis a Bayesian approach to parameter estimation and associate the rule parameters with explicit prior distributions over their range of possible values. The benefits of Bayesian approaches compared to traditional maximum likelihood methods are multiple:

1. Bayesian methods allow domain knowledge to be included into the learning cycle through the use of informative priors (cf. discussion below). The system designer is thus free to bias the initial prior distributions according to her/his domain expertise.

2. Bayesian methods explicitly capture the model uncertainty both before and after learning. The outputs of a Bayesian learning cycle are indeed full posterior distributions over the possible parameter values instead of being reduced to point estimates (as for maximum likelihood estimation). The dialogue agent can therefore explicitly account for parameter uncertainty at runtime and use it to simultaneously learn, reason and act in its environment (Ross et al., 2011). Furthermore, the reliance on full parameter distributions facilitates the combination of multiple learning processes, as the posterior distribution generated by one learner can be passed on as the prior distribution of another.

Prior parameter distributions have a continuous range of values and can be either univariate or multivariate. We first review prior parameter distributions for probability rules and then discuss the case of utility rules.

Probability parameters

The distributions over the effects of probability rules are categorical probability distributions. As discussed in Section 3.1.3, the parameter priors of categorical and multinomial distributions are best expressed using *Dirichlet* distributions.

Each distribution over effects $P(E_i)$ is defined by its own Dirichlet distribution of dimension m_i, where m_i is the number of alternative effects (including the empty effect if appropriate). Rule r_1 provides an example of parametrised probability rule:

$$
\begin{aligned}
r_1: \quad & \textbf{if } (Rain = false \wedge Weather = hot) \textbf{ then} \\
& \begin{cases} P(Fire = true) = \theta_{r_1(1,1)} \\ P(Fire = false) = \theta_{r_1(1,2)} \end{cases} \\
& \textbf{else} \\
& \begin{cases} P(Fire = true) = \theta_{r_1(2,1)} \\ P(Fire = false) = \theta_{r_1(2,2)} \end{cases}
\end{aligned}
$$

The parameters of rule r_1 can be expressed with two independent Dirichlet distributions $P(\boldsymbol{\theta}_{r_1(1,\cdot)})$ and $P(\boldsymbol{\theta}_{r_1(2,\cdot)})$ respectively associated with the effect distributions for the first and second condition. The Dirichlet distributions in this example each have two dimensions (since two alternative effects are mentioned in the rule), which make them equivalent to *Beta* distributions.

Dirichlet distributions are continuous, multivariate distributions defined by the meta-parameters $\boldsymbol{\alpha} = [\alpha_1, ...\alpha_k]$, where k corresponds to the dimensionality of the categorical distribution of interest. Dirichlet distributions over the parameters $\theta_1, ...\theta_k$ are formally expressed by the following probability density function:

$$p(\theta_1, ...\theta_k \,;\boldsymbol{\alpha}) = \frac{1}{\mathrm{B}(\boldsymbol{\alpha})} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1} \quad \text{where } \mathrm{B}(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)} \tag{5.3}$$

where $\mathrm{B}(\boldsymbol{\alpha})$ serves as normalisation factor and builds upon on the gamma function Γ.[2] We remind the reader that the notation $p(\theta_1, ...\theta_k \,;\boldsymbol{\alpha})$ refers to the density function of the $\theta_1, ...\theta_k$ random variables given the specified (hyper-)parameters $\boldsymbol{\alpha}$.

Figure 5.1 illustrates the shape of the probability density functions for several variants of the two-dimensional Dirichlet distribution $P(\theta_1, \theta_2 \,;\boldsymbol{\alpha})$. The second dimension θ_2 is not explicitly shown on the figure but can be directly derived from the first dimension, since $\theta_1 + \theta_2 = 1$. We can observe how the $\boldsymbol{\alpha}$ hyper-parameters determine the shape of the distribution. As the $\boldsymbol{\alpha}$ counts grow larger, the density function becomes increasingly focused on a particular region of the parameter space. The $\boldsymbol{\alpha}$ counts can therefore be tuned to skew the distribution in a particular way based on prior domain knowledge. Such prior distributions are called "informative" priors, since their shape is influenced by expert information. In the absence of such information, non-informative distributions such as $\mathrm{Dirichlet}(1, 1)$ can also be employed.

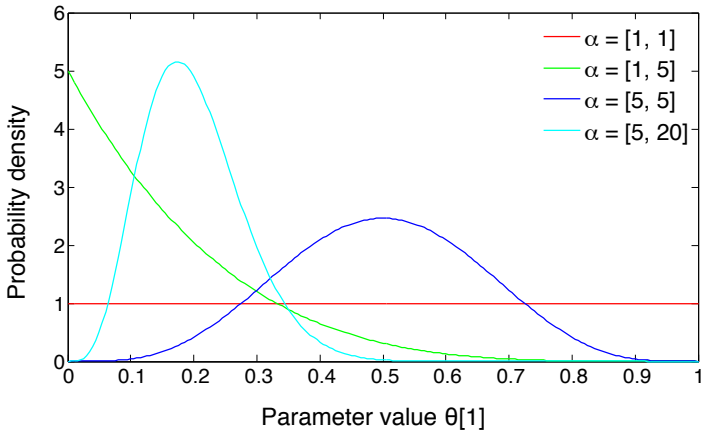


Figure 5.1: Probability density functions for the Dirichlet distribution $p(\theta_1, \theta_2 \,;\boldsymbol{\alpha})$ with various values for the $\boldsymbol{\alpha}$ hyper-parameters.

As Dirichlet distributions are conjugate priors of categorical distributions, their posterior distribution after the observation of their corresponding variable remains a Dirichlet distribution with updated counts. As an illustrative example, the posterior distribution of $\boldsymbol{\theta}_{r_1(1,\cdot)}$ after observing a fire when $\mathit{Rain} = \mathit{false} \wedge \mathit{Weather} = \mathit{hot}$ can be derived from Bayes' rule:

$$\begin{aligned} &P(\boldsymbol{\theta}_{r_1(1,\cdot)} \mid \mathit{Fire} = \mathit{true}, \mathit{Rain} = \mathit{false}, \mathit{Weather} = \mathit{hot}) \\ &= \eta \, P(\mathit{Fire} = \mathit{true} \mid \mathit{Rain} = \mathit{false}, \mathit{Weather} = \mathit{hot}, \boldsymbol{\theta}_{r_1(1,\cdot)}) \, P(\boldsymbol{\theta}_{r_1(1,\cdot)}) \end{aligned}$$

[2] The gamma function is a generalisation of the factorial for real numbers, and is defined as $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} \, \mathrm{d}t$.

$$= \eta\, \theta_{r_1(1,1)}\, \text{Dirichlet}(\alpha_1, \alpha_2)$$
$$= \eta'\, \theta_{r_1(1,1)}\, [\theta_{r_1(1,1)}^{\alpha_1 - 1} \times \theta_{r_1(1,2)}^{\alpha_2 - 1}] = \eta'\, \text{Dirichlet}(\alpha_1 + 1, \alpha_2)$$

where η, η' are normalisation factors. Given a prior distribution $P(\boldsymbol{\theta}_{r_1(1,\cdot)}) \sim \text{Dirichlet}(\alpha_1, \alpha_2)$, the posterior distribution for $\boldsymbol{\theta}_{r_1(1,\cdot)}$ after the observation $Fire = true$ is thus another Dirichlet distribution $\sim$ $\text{Dirichlet}(\alpha_1 + 1, \alpha_2)$. As explained in Section 3.1.3, this property is however contingent on the full observability of the domain variables.

Utility parameters

The parameters of utility rules are also defined by probability density functions. However, contrary to probability values, the values in a utility distribution are independent of one another and need not satisfy the probability axioms (their range of possible values is arbitrary). Each utility value $u_{(i,j)}$ assigned to decision $d_{(i,j)}$ is therefore associated with its own, univariate distribution.

Rule r_2 illustrates a utility rule with four independent parameters:

$$r_2 : \textbf{ if } (Fire = true) \textbf{ then}$$
$$\begin{cases} U(\,Tanker = drop\text{-}water) = \theta_{r_2(1,1)} \\ U(\,Tanker = wait) = \theta_{r_2(1,2)} \end{cases}$$
$$\textbf{else}$$
$$\begin{cases} U(\,Tanker = drop\text{-}water) = \theta_{r_2(2,1)} \\ U(\,Tanker = wait) = \theta_{r_2(2,2)} \end{cases}$$

Several types of density functions can be applied to define the prior distributions over these utility values. This thesis concentrates on two specific families of priors, one non-informative (uniform distributions) and one informative (normal distributions):

1. Continuous uniform distributions are defined on an interval $[a, b]$ which corresponds to the allowed range of utility values, and have the following density:

$$p(\theta\,; a, b) = \begin{cases} \frac{1}{b-a} & \text{for } \theta \in [a, b] \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

2. Normal (also called Gaussian) distributions are defined by a probability density function revolving around a mean μ and variance σ^2:

$$p(\theta\,; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{(\theta - \mu)^2}{2\sigma^2} \right\} \tag{5.5}$$

   The range of possible values can be further constrained by truncating the density function.

Normal distributions are well suited to represent utility values for which rough initial estimates are available. If a particular utility value is expected to lie in the vicinity of a particular value $\tilde{u}$, its probability distribution can be expressed via a normal distribution with a mean $\mu = \tilde{u}$ and a variance σ^2 reflecting the confidence in the provided estimate.

Figure 5.2 illustrates three instances of probability density functions for a parameter θ. The first distribution corresponds to a uniform distribution on the interval $[-2, 4]$, while the second and third distributions are truncated normal distributions with mean $\mu = 2$ and variances respectively assigned to $\sigma^2 = 4$ and $\sigma^2 = 1$. The normal distributions illustrate how prior knowledge about the utility value can be incorporated in the prior – in this case, the distributions rest on the assumption that the true utility value is likely to revolve around the value 2.

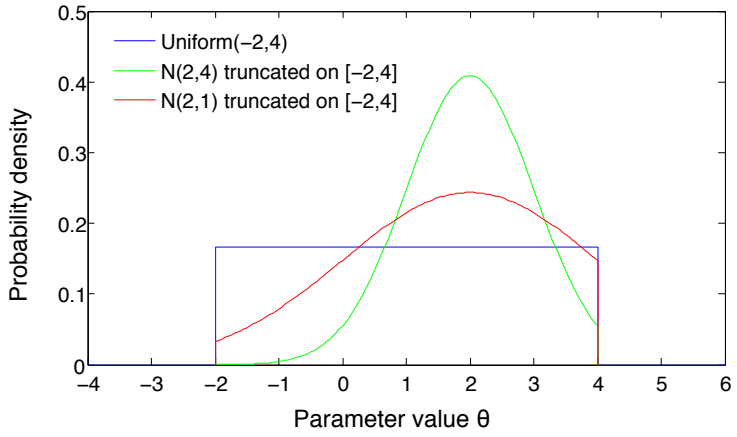


Figure 5.2: Probability density functions $P(\theta)$ over the interval $[-2, 4]$ using an uniform distribution, a truncated normal distribution $\mathcal{N}(2, 4)$ and a truncated normal distribution $\mathcal{N}(2, 1)$.

5.1.3 Instantiation

Parameters are instantiated in the dialogue state as distinct chance nodes. One chance node is created for each (univariate or multivariate) parameter distribution and included as parents of its corresponding rule node. The rule distributions must be slightly adapted to factor these parameters in the parent nodes of the rule. Other aspects of the instantiation process – such as output distributions or quantifiers – remain unchanged.

Parameters of probability rules

A parametrised probability rule r structured with n conditions is associated with n multivariate parameter nodes $\boldsymbol{\theta}_{r(1,\cdot)}, ...\boldsymbol{\theta}_{r(n,\cdot)}$. Each distribution $P(\boldsymbol{\theta}_{r(i,\cdot)})$ is a Dirichlet distribution of dimension m_i, where m_i is the number of effects associated with the condition c_i (including the empty effect). Figure 5.3 illustrates this instantiation procedure with two probability rules.

The conditional probability distribution of a rule node r given its input variables $I_1, ...I_k$ and parameters $\boldsymbol{\theta}_{r(1,\cdot)}, ...\boldsymbol{\theta}_{r(n,\cdot)}$ is a straightforward adaptation of Equation (4.4):

$$P(r = e \mid I_1 = i_1, ...I_k = i_k\,; \boldsymbol{\theta}_{r(1,\cdot)}, ...\boldsymbol{\theta}_{r(n,\cdot)}) = P(E_i = e\,; \boldsymbol{\theta}_{r(i,\cdot)}) \qquad (5.6)$$
$$\text{where } i = \min_i(c_i \text{ is satisfied with } I_1 = i_1 \wedge ...I_k = i_k)$$

Parameters of utility rules

The parameters of utility rules are instantiated in a similar manner, as shown in Figure 5.4. The corresponding utility distribution is adapted from Equation 5.7 as follows:

$$\begin{aligned} &U_r(I_1 = i_1, ...I_k = i_k, A_1 = a_1, ...A_l = a_l\,; \boldsymbol{\theta}_{r(1,\cdot)}...\boldsymbol{\theta}_{r(n,\cdot)}) \\ &= U_i(A_1 = a_1, ...A_l = a_l\,; \boldsymbol{\theta}_{r(i,\cdot)}) \\ &\qquad \text{where } i = \min_i(c_i \text{ is satisfied with } I_1 = i_1 \wedge ...I_k = i_k) \end{aligned} \tag{5.7}$$

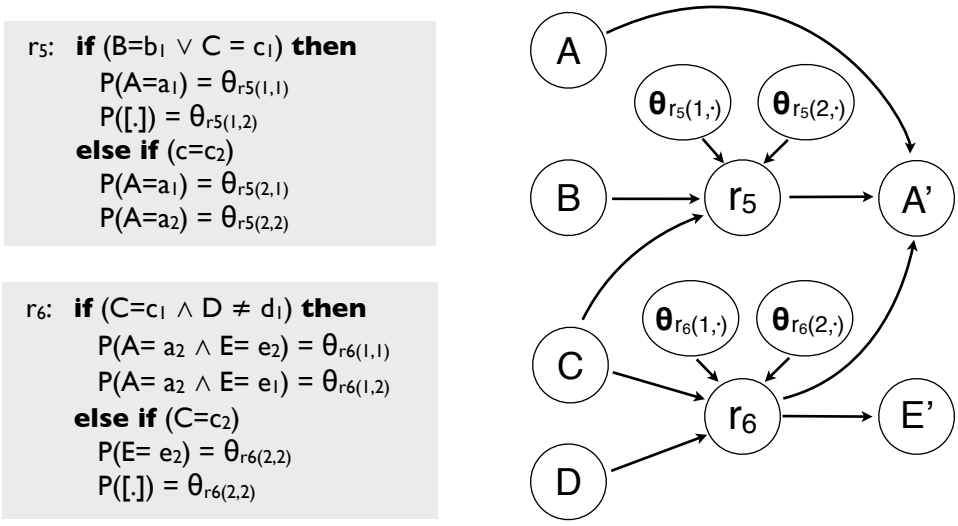


Figure 5.3: Example of instantiation for two parametrised probability rules r_5 and r_6.

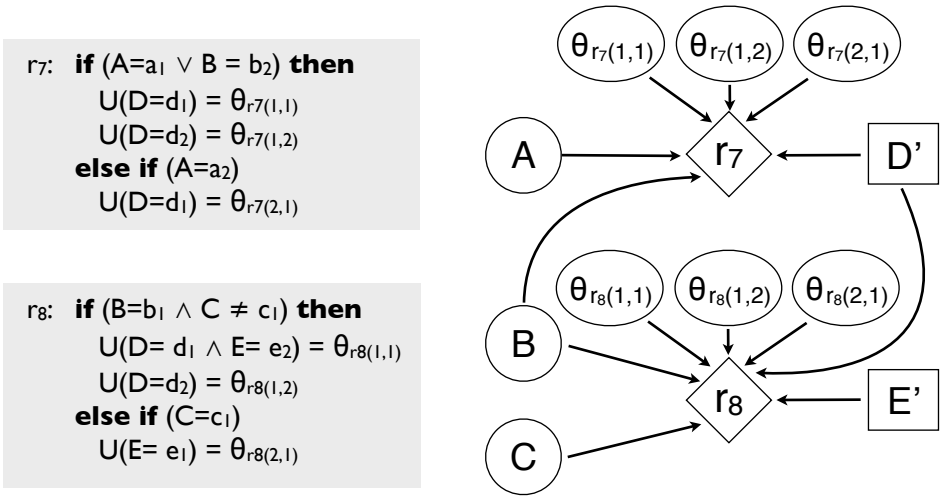


Figure 5.4: Example of instantiation for two parametrised utility rules r_7 and r_8.

5.2 Supervised learning of rule parameters

Once the rule parameters are instantiated as nodes in the dialogue state, one can calculate their posterior distribution $P(\boldsymbol{\theta} \mid \mathcal{D})$ after observing a particular data set $\mathcal{D}$ using standard algorithms for probabilistic inference. The estimation of rule parameters corresponds to a learning problem with partial data (cf. Section 3.1.3), since a subset of chance variables – such as rule nodes – are not directly observed. The posterior distribution $P(\boldsymbol{\theta} \mid \mathcal{D})$ is thus no longer guaranteed to remain in the same distribution family as their prior distributions. The solution adopted in this thesis is to approximate the parameter distributions via sampling techniques, as shall be explained below.

The estimation of rule parameters from data is practically achieved by cycling through the data sample one after the other and gradually refining the parameter distributions on their basis. We describe in the next pages the generic representation of the training data, the as well as the learning algorithm that exploits them for optimising the rule parameters of the domain given some reasonable assumptions about the wizard behaviour.

5.2.1 Wizard-of-Oz training data

This chapter focus on parameter estimation from a specific type of training data, namely Wizard-of-Oz interactions. Wizard-of-Oz interactions are interactions between human users and a dialogue system which is remotely controlled by a human expert "behind the curtains". They constitute a simple and efficient method to collect realistic conversational behaviour for a particular domain in the absence of a fully implemented or optimised system.[3]

Representation format

For the specific purpose of estimating the parameters of dialogue management models, we can represent Wizard-of-Oz interactions as a sequence of state-action pairs $\mathcal{D} = \{(\mathcal{B}_i, a_i) : 1 \leq i \leq n\}$, where $\mathcal{B}_i$ corresponds to the dialogue state at time i, and a_i is the associated action performed by the wizard. The number n corresponds to the total number of recorded actions.

The dialogue state $\mathcal{B}_i$ represents the current conversational situation at time i as it was perceived by the wizard. Its representation usually includes the recent dialogue history as well as important contextual features. As explained in the previous chapter, the dialogue state can be encoded as a Bayesian network to reflect state uncertainty and dependences amongst state variables. Associated to each dialogue state is the corresponding action a_i selected by the wizard at that state. This action can be void if the wizard decides to take no action at that specific step in the dialogue.

Many conversational situations allow for multiple, equally "correct" system responses. This characteristic of verbal interactions transpires in the state-action pairs of Wizard-of-Oz data sets, as one can occasionally observe similar states mapped to different wizard actions. The wizard actions should therefore be viewed as an indication of good conversational behaviour, but do not constitute absolute gold standards in the traditional sense of being the uniquely appropriate output for the given dialogue state. The existence of multiple responses also entails that the accuracy of the learned models remains contingent on the degree of internal consistency of the wizard actions.

[3] Developing spoken dialogue systems can indeed lead to a classical "chicken-and-egg" dilemma: in order to build up a particular system, system designers often need to know what types of user utterances and behaviours are expected – but in order to collect such data, one must first have an integrated dialogue system with which the users can interact. Wizard-of-Oz interactions are a way to circumvent this dilemma.

Assumptions about the wizard behaviour

Parameter estimation on Wizard-of-Oz data rests on the assumption that the wizard is a rational agent and will tend to select actions that are deemed most useful in their respective dialogue state. It should be stressed that the agent is only assumed to act rationally *given* the perceived (uncertain) dialogue state. The wizard must indeed act on the basis of "noisy" inputs (including e.g. speech recognition errors) and may as a consequence select suboptimal actions if the provided inputs contain erroneous hypotheses. It should hence be stressed that the assumption of rationality does not equate to an assumption of omniscience on the part of the wizard.

Practically, the presupposed rationality of the wizard implies that the likelihood $P_{\mathcal{B}_i}(a_i\,;\boldsymbol{\theta})$ of a wizard action a_i in a particular dialogue state $\mathcal{B}_i$ under the parameters $\boldsymbol{\theta}$ will be proportional to the corresponding utility of action a_i in $\mathcal{B}_i$ relative to other actions (as formalised below).

5.2.2 Learning cycle

The goal of the learning process is to estimate the posterior distribution $P(\boldsymbol{\theta}\,|\,\mathcal{D})$ over the rule parameters given the collected Wizard-of-Oz data set. The procedure operates in an incremental fashion by traversing the (state,action) pairs one by one and re-estimating the posterior distribution after each pair.

Likelihood distribution

One key element of the learning cycle is the definition of the probability distribution $P_{\mathcal{B}_i}(a_i\,;\boldsymbol{\theta})$, which specifies the likelihood of the wizard action a_i in a dialogue state $\mathcal{B}_i$ given the parameters $\boldsymbol{\theta}$. The purpose of this likelihood is intuitively to favour the parameter values that provide a good fit for the wizard action choices. This is practically achieved by defining the likelihood as the relative utility of the action a_i compared to all other actions:

$$P_{\mathcal{B}_i}(a_i\,;\boldsymbol{\theta}) \leftarrow \frac{U_{\mathcal{B}_i}(a_i\,;\boldsymbol{\theta}) - U_{\min}}{\sum_{a\in Val(A)}\left(U_{\mathcal{B}_i}(a\,;\boldsymbol{\theta}) - U_{\min}\right)} \tag{5.8}$$

Optimal parameter values with respect to the wizard action will therefore assign a high utility to the action a_i and low utilities to the other actions. Equation (5.8) includes a minimal utility threshold $U_{\min}$ to ensure that the likelihood remains positive. This minimal threshold can be either set manually by the system designer (which is the approach taken in our experiment) or automatically derived from the domain models.

Posterior parameter distribution

Given the aforementioned likelihood distribution, the posterior distribution over the parameters is defined via Bayes' rule:

$$P_{\mathcal{B}_i}(\boldsymbol{\theta}\,|\,a_i) = \eta\, P_{\mathcal{B}_i}(a_i\,;\boldsymbol{\theta})\, P(\boldsymbol{\theta}) \tag{5.9}$$

The factor η is used for normalisation. The value range for parameter distributions is typically continuous and contains as a consequence an infinite number of possible values. Inference in hybrid graphical models featuring both continuous and discrete variables is known to be a non-trivial problem. Two types of solutions can be distinguished:

- The first strategy is to discretise the range of parameter values into distinct, mutually exclusive buckets, and thereby transform continuous variables into discrete variables, with a number of values equivalent to the number of buckets employed for the discretisation.

- The second strategy is to retain the continuous nature of the parameters, but approximate the inference process through the use of sampling techniques.

Although both solutions are implemented in the `openDial` toolkit, sampling techniques such as likelihood weighting have proved to be in practice more efficient and scalable than discretisation.

After sampling, the full joint distribution $P_{\mathcal{B}_i}(\boldsymbol{\theta} \mid a_i)$ is factored into its individual parameter variables. This factoring is a simplifying assumption, since parameter independence is theoretically no longer guaranteed when handling partially observed data.

Representation of the posterior

The result of the posterior calculation in Equation (5.9) is a collection of sampled values. Two approaches are possible to define probability density functions from these samples. One can either follow a so-called parametric approach and seek to reconstruct the underlying parametric distributions (such as Gaussian or Dirichlet distributions) that best fit the data, or adopt a non-parametric approach and directly represent the posterior as a function of the samples. Parametric approaches, albeit interesting, are difficult to apply in this setting, for two main reasons:

1. The distribution family of the posterior is hard to determine, as the posterior is no longer ensured to remain in the same family as the prior when learning from partial data.

2. Additionally, fitting multivariate distributions such as Dirichlets based on sampled values is a laborious computational process with no closed-form solutions.[4]

We have consequently adopted a non-parametric representation of the posterior distributions, based on *Kernel Density Estimation* (KDE). The kernel density estimator for a continuous variable X for which a set of samples $x_1, ...x_n$ is available is given by:

$$p(x) = \frac{1}{nh} \sum_{i=1}^{n} K\Big(\frac{x - x_i}{h}\Big) \tag{5.10}$$

where $K(\cdot)$ is a *kernel function* and h is a smoothing parameter called the *bandwidth*. Multiple kernel functions can be used, but a common choice is to adopt a Gaussian kernel. The kernel density estimator corresponds in this case to a combination of n Gaussians, where each Gaussian is centered on a sample point x_i. This combination of Gaussians is smoothed proportionally to the bandwidth parameter. Figure 5.5 illustrate the use of kernel density estimation for a continuous variable based on a set of 50 samples and a Gaussian kernel. The figure shows the influence of the bandwidth parameter on the shape of the resulting density function. Kernel density estimators do not necessitate any particular assumption about the nature of the underlying distribution, and are therefore well-suited to represent distributions of indeterminate type – as is the case for posterior distributions over the parameters of probabilistic rules.

[4] Numerical methods based on fixed-point and Newton-Raphson iterations do however exist (Minka, 2003).

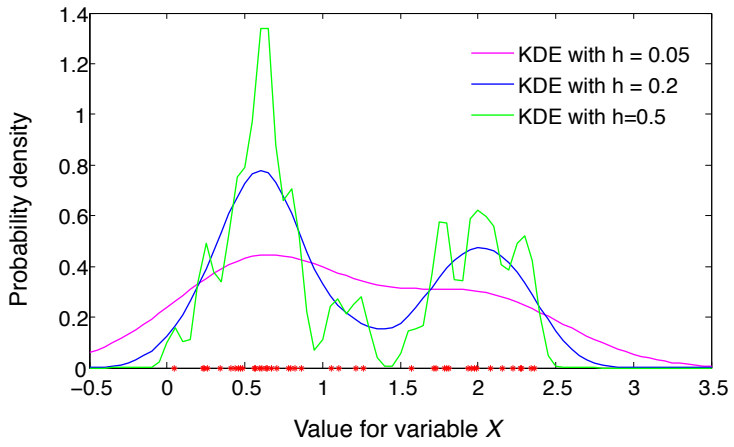


Figure 5.5: Kernel density estimators (KDEs) for a continuous variable X based on 50 samples (shown on the X axis). The density function is shown for three possible bandwidths h.

Multivariate distributions (arising from Dirichlet priors) are encoded with a multivariate extension of kernel density estimators based on product kernels (Silverman, 1986). All estimators in our experiments rely on Gaussian kernels with a bandwidth tuned from the sample variance using Silverman's rule of thumb (Silverman, 1986).

Learning algorithm

Algorithm 12 presents the general procedure for estimating model parameters from Wizard-of-Oz data. The algorithm loops on each instance pair in the training data. For each pair, the algorithm starts by including the parameters in the dialogue state and triggering the domain models (line 2 and 3). The posterior parameter distribution is then estimated via sampling, based on the wizard action likelihood (line 4). The posterior distribution for each parameter is finally expressed as a kernel density estimator on the sampled values (line 5-7), and the process is repeated.

Algorithm 12 : WOZ-LEARNING $(\mathcal{M}, \boldsymbol{\theta}, \mathcal{D}, N)$

Input: Rule-structured models $\mathcal{M}$ for the domain
Input: Model parameters $\boldsymbol{\theta}$ with prior distribution $P(\boldsymbol{\theta})$
Input: Wizard-of-Oz data set $\mathcal{D} = \{\langle \mathcal{B}_i, a_i \rangle : 1 \leq i \leq n\}$
Input: Number N of samples to draw for each learning example
Output: Posterior distribution $P(\boldsymbol{\theta} \mid \mathcal{D})$ for the parameters

1: **for all** $\langle \mathcal{B}_i, a_i \rangle \in \mathcal{D}$ **do**
2: Set $\mathcal{B}_i \leftarrow \mathcal{B}_i \cup \boldsymbol{\theta}$
3: $\mathcal{B}_i, \mathrm{e} \leftarrow$ TRIGGERMODELS $(\mathcal{B}_i, \emptyset, \mathcal{B}_i)$
4: Draw N samples $\mathbf{x}_1, ...\mathbf{x}_N$ from posterior $P_{\mathcal{B}_i}(\boldsymbol{\theta} \mid a_i) = \eta \, P_{\mathcal{B}_i}(a_i \,; \boldsymbol{\theta}) \, P(\boldsymbol{\theta})$
5: **for all** parameter variable $\theta \in \boldsymbol{\theta}$ **do**
6: Set $P(\theta) \leftarrow \mathrm{KDE}(\mathbf{x}_1(\theta), ...\mathbf{x}_N(\theta))$
7: **end for**
8: **end for**
9: **return** $P(\boldsymbol{\theta})$

5.3 Experiments

We evaluated the learning approach outlined in this chapter in the context of a dialogue policy learning task for a human-robot interaction scenario. The goal of the experiment, originally presented in Lison (2012d), was to evaluate how well the parameters of a rule-structured utility model could be optimised from small amounts of Wizard-of-Oz data. The evaluation metric was defined in this experiment as the proportion of actions corresponding to the wizard selections. The rule-structured model was compared to two baselines where the action utilities was represented using classical representations (respectively utility tables and linear functions).

It should be stressed that the purpose of the experiment is limited to the evaluation of the *learning* performance of the model. The evaluation of the model in terms of e.g. qualitative and quantitative metrics of interaction success (and user satisfaction) constitutes an important but separate question, which will be addressed in Chapter 8.

We first describe the dialogue domain used for the experiment, after which we detail the data collection procedure and experimental setup, and finally present and analyse the empirical results.

5.3.1 Dialogue domain

The scenario for the Wizard-of-Oz experiment involved a human user and a Nao robot (nicknamed "Lenny"), which is a programmable humanoid robot developed by Aldebaran Robotics. Figure 5.6 shows a human user interacting with the robot during a data collection experiment.

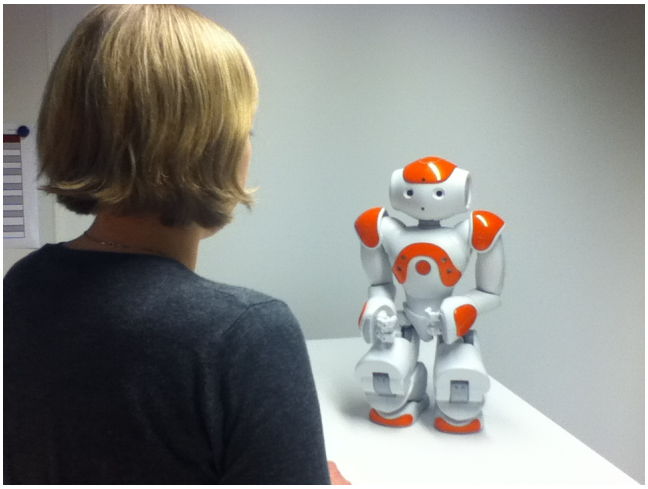


Figure 5.6: Human user interacting with the Nao robot during the Wizard-of-Oz data collection.

The users were instructed to teach the robot a sequence of body movements such as lifting arms, stepping forward, backward, kneeling down, etc. The movements could be performed either in consecutive order or in parallel (provided the movements were not in conflict). The users were free to decide on the movements to perform, and communicated their intention using spoken commands (no gesture recognition was here involved). The robot was programmed to memorise the instructed sequence and could "replay" it at any time.

The list of possible dialogue acts for the user is shown in Table 5.1, and includes a total of 16 dialogue act templates (expanding into 41 dialogue acts when counting all possible argument

instantiations). The set of user dialogue acts contain both task-specific dialogue moves to convey user commands as well as conversational actions for feedbacks, acknowledgements, corrections and engagement. To respond to these user inputs, the robot/wizard had at its disposal a repository of 12 possible actions (expanding into 41 alternative actions when counting all possible instantiations of the action arguments). The actions included both physical and verbal actions. The verbal actions available to the system comprised various types of clarification requests and grounding acts. The list of system actions is given in Table 5.2.

- MoveArm(x, y)
  where $x = \{\text{Left}, \text{Right}, \text{Both}\}$
  and $y = \{\text{Up}, \text{Down}, \text{Lateral}, \text{Forward}, \text{Folded}\}$
- MoveHead(y)
  where $y = \{\text{Up}, \text{Left}, \text{Down}, \text{Right}\}$
- MoveFoot(x, y)
  where $x = \{\text{Left}, \text{Right}\}$
  and $y = \{\text{Forward}, \text{Backward}\}$
- Turn(y)
  where $y = \{\text{Left}, \text{Right}\}$
- Kneel
- StandUp
- SitDown
- DoMovements(y)
  where $y = \{\text{InParallel}, \text{InSequence}\}$
- RepeatAll
- ForgetAll
- Confirm
- Disconfirm
- Say(x)
  where $x = \{\text{Hello}, \text{Compliment}, \text{ThankYou}, \text{Goodbye}\}$
- GoToInitPose
- FollowMe
- Stop

Table 5.1: List of user actions a_u

- Demonstrate(z)
  where $z = \{\text{MoveArm}(x, y), \text{MoveHead}(y), \text{Kneel}, \text{StandUp}, \text{SitDown}, \text{MoveFoot}(x, y), \text{Turn}(y)\}$
  and x, y take the same values as for the user actions
- Say(x)
  where $x = \{\text{Hello}, \text{ThankYou}, \text{Goodbye}\}$
- AskConfirmation
- RegisterMove
- UndoMove
- AskRepeat
- Acknowledgement
- AskIntention
- DemonstrateAll
- ForgetAll
- StopMove
- FollowUser

Table 5.2: List of system actions a_m

5.3.2 Wizard-of-Oz data collection

System platform

An integrated dialogue system was developed in order to collect Wizard-of-Oz interactions for the human-robot interaction domain described above. The dialogue system was equipped with all standard processing modules for speech understanding, generation and robot control. Figure 5.7 illustrates the general system architecture and its connection to the robotic platform. As evidenced

by the figure, the general system architecture is directly inspired by information-state approaches. At the centre of the architecture lies a shared dialogue state to which multiple system components are attached. These components monitor the dialogue state for relevant changes and read/write to it as they process their data flow. The dialogue management module is naturally replaced by the wizard during data collection.

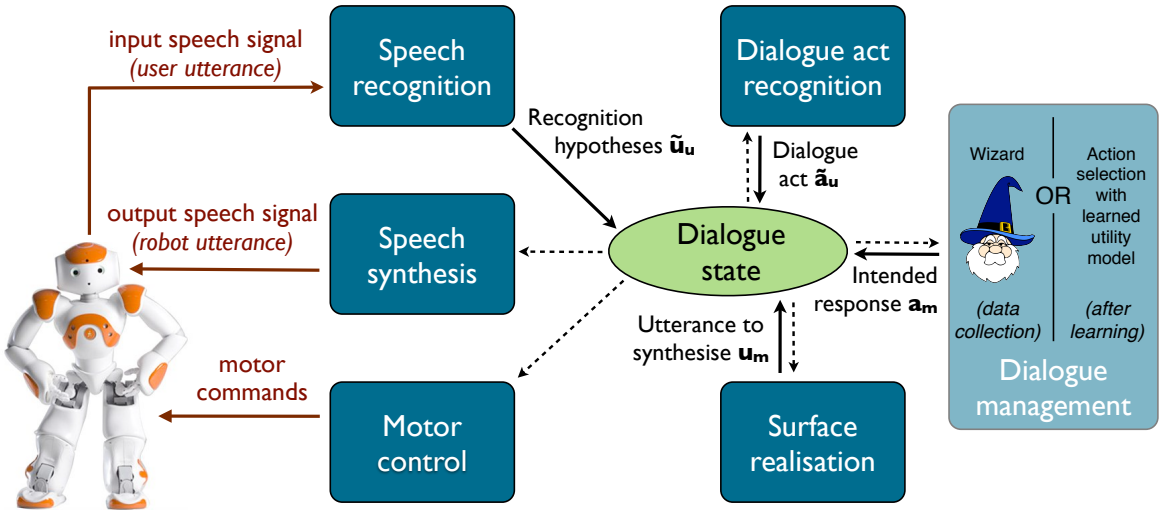


Figure 5.7: System architecture used for the experiment.

The modules used for the experiments included in particular an off-the-shelf speech recogniser (Vocon 3200 from Nuance) connected to four microphones placed on the robot head. The language model of the speech recogniser was represented by a small hand-crafted recognition grammar, while the acoustic model was set to U.S. English. The corresponding user dialogue acts were then derived from the ASR hypotheses via a template-based recognition model. On the generation side, a shallow generation model was in charge of the surface realisation of the verbal actions. The surface string was synthesised via a speech synthesis module embarked on the robot. Finally, the execution of physical actions was delegated to a separate component, responsible for planning the robot movements and controlling its motors in real-time, based on the software libraries available on the robotic platform.

Data collection procedure

Each recorded dialogue involved one human subject interacting with the Nao robot in a shared visual scene. The dialogues were relatively short, with an average duration of about four minutes. We collected a total of 20 interactions with 7 distinct users, for a total of 1020 system turns, summing to around 1h of interaction. The users were recruited amongst the students and researchers in the Department of Informatics at the University of Oslo, while the role of the wizard was taken by the author of the present thesis. All the interactions were performed in English. All users were comfortable speaking English, but all but one were non-native speakers.[5] As the speech recogniser

[5]This may partly explain the relatively high level of speech recognition errors due to the mismatch between the acoustic model trained on native American speakers and the foreign accents of the participants.

relied on a grammar-based language model of limited size, the users were briefed before each experiment about the comprehension capabilities of the robot in order to adjust their expectations about what the robot could and could not understand, and thereby limit the number of out-of-coverage utterances.

Wizard-of-Oz experiments should ideally place the wizard in the same types of decision contexts as the ones encountered by the dialogue manager. To this end, the wizard was prevented from listening directly to the spoken utterances of the user and was instead provided with the N-best list $\tilde{a}_u$ generated by the dialogue act recognition engine (which is itself based on the speech recogniser outputs). The N-best lists appeared on the wizard control screen as lists of hypotheses accompanied with their respective probabilities. On the basis of these inputs (and the interaction context), the wizard could then select the action to perform from a list of alternatives.

Transcripts 1 and 2 present two recorded excerpts of Wizard-of-Oz interactions. The user utterances are displayed as N-best lists of speech recognition hypotheses.

Dialogue state

Each selected action a_m was recorded along with the complete dialogue state $\mathcal{B}$ in effect at the time of the selection. The dialogue state variables are represented with their full probability distributions.

The dialogue state designed for the experiment consisted of five independent variables:

1. The last user dialogue act a_u (with the values given in Table 5.1)
2. The last system action a_m (with the values given in Table 5.2)
3. The recorded sequence of (confirmed) movements, encoded as a list (cf. Section 4.5)
4. The last physical movement demonstrated by the robot
5. Finally, a periodically updated variable expressing the number of seconds elapsed since the last user or system action. This variable is used to determine when the system should ask for an explicit confirmation from the user.

The dialogue state recorded right before the wizard registers the first movement in Transcript 1 corresponds for instance to the following specification:

$$\mathcal{B} = \begin{cases} a_u = \langle(\text{Confirm}, p{=}0.88), (\text{SayThankYou}, p{=}0.10), (\mathit{None}, p{=}0.02)\rangle \\ a_m = \langle(\text{AskConfirmation}, p{=}1)\rangle \\ \mathit{moveSequence} = \langle(\emptyset, p{=}1)\rangle \\ \mathit{lastMove} = \langle(\text{MoveArm}(\text{Left}, \text{Up}), p{=}1)\rangle \\ \mathit{silenceTime} = \langle(3 \text{ seconds}, p{=}1)\rangle \end{cases}$$

5.3.3 Experimental setup

The central question investigated in this experiment is the following: does the encoding of dialogue management models in terms of probability and utility rules really improve the efficiency of the parameter estimation process compared to more traditional representations? And if yes, how significant is the difference? The experiment focused therefore on the learning performance of various

Transcript 1 User interaction with wizard-controlled robot, first excerpt

HUMAN:
$$\begin{bmatrix} \text{hello Lenny} & 0.22 \\ \text{follow me} & 0.20 \\ \text{hi Lenny} & 0.53 \end{bmatrix}$$

ROBOT: Hello there!

HUMAN:
$$\begin{bmatrix} \text{move your left arm up} & 0.41 \\ \text{move the left arm up} & 0.27 \\ \text{put your left arm up} & 0.30 \end{bmatrix}$$

ROBOT: Sure!

[ROBOT *executes the movement* $\mathrm{MoveArm}(\mathrm{Left}, \mathrm{Up})$]
[ROBOT *waits a few seconds*]

ROBOT: Was that correct?

HUMAN:
$$\begin{bmatrix} \text{yes} & 0.88 \\ \text{thanks} & 0.10 \end{bmatrix}$$

[ROBOT *registers the movement as confirmed*]

ROBOT: Great

HUMAN:
$$\begin{bmatrix} \text{move your right arm up at the same time} & 0.26 \\ \text{put the right arm up at the same time} & 0.32 \\ \text{move the right arm up at the same time} & 0.38 \end{bmatrix}$$

ROBOT: Let's see

[ROBOT *executes movement* $\mathrm{MoveArm}(\mathrm{Both}, \mathrm{Up})$]

ROBOT: Was that correct?

HUMAN:
$$\begin{bmatrix} \text{kneel} & 0.25 \\ \text{thanks} & 0.14 \\ \text{no} & 0.57 \end{bmatrix}$$

[ROBOT *discards last movement and goes back to previous position*]

ROBOT: What should I do then?

[...]

Transcript 2 User interaction with wizard-controlled robot, second excerpt

[...]

HUMAN:
$$\begin{bmatrix} \text{stop} & 0.25 \\ \text{stand up} & 0.41 \\ \text{no} & 0.25 \end{bmatrix}$$

ROBOT: Let's see

[ROBOT *executes movement* StandUp]

HUMAN:
$$\begin{bmatrix} \text{put your right arm forward} & 0.31 \\ \text{move your right arm forward} & 0.36 \\ \text{move the right arm forward} & 0.30 \end{bmatrix}$$

[ROBOT *registers movement* StandUp *as implicitly confirmed*]

ROBOT: Let's see

[ROBOT *executes movement* MoveArm(Right, Forward)]

HUMAN:
$$\begin{bmatrix} \text{move forward the left arm} & 0.31 \\ \text{lower the left arm} & 0.35 \\ \text{lower both arms} & 0.29 \end{bmatrix}$$

ROBOT: Sorry, could you repeat?

HUMAN:
$$\begin{bmatrix} \text{raise your left arm} & 0.28 \\ \text{lift your right arm} & 0.31 \\ \text{lift your left arm} & 0.37 \end{bmatrix}$$

ROBOT: Sure

[ROBOT *executes movement* MoveArm(Left, Up)]

[ROBOT *waits a few seconds*]

ROBOT: Should I do it like this?

HUMAN:
$$\begin{bmatrix} \text{kneel} & 0.23 \\ \text{wrong} & 0.22 \\ \text{no} & 0.47 \end{bmatrix}$$

[ROBOT *discards last movement and goes back to previous position*]

ROBOT: What should I do then?

HUMAN:
$$\begin{bmatrix} \text{move your left arm up} & 0.29 \\ \text{move the left arm up} & 0.34 \\ \text{move the left arm down} & 0.28 \end{bmatrix}$$

[...]

utility models based on limited amounts of training data gathered from Wizard-of-Oz interactions. The parameters correspond here to the utilities of the various system actions depending on the current state.

Baseline models

The experiment relied on two distinct baselines that express the utility model of the domain based on traditional representations:

1. The first baseline is a plain utility table that maps every combination of state values and actions to a particular utility. For a given set of state variables $X_1, ...X_n$, the utility for the system action a'_m is therefore defined by:

$$U(X_1 = x_1, ...X_n = x_n, a'_m) = \theta_{(x_1,...x_n,a'_m)} \tag{5.11}$$

   where the $\theta_{(x_1,...x_n,a'_m)}$ value corresponds to the utility encoded in the table for the state-action pair. The total number of required parameters is therefore $|Val(X_1)|\times...|Val(X_n)|\times |Val(a'_m)|$. In order to keep the model tractable, the utility table was factored in the experiments in three parts, each responsible for a subset of the possible system actions. These utility tables comprised a total of 8962 independent parameters.

2. The second baseline defines the utility of a given action as a linear combination of values – one for each state variable. The total utility is thus determined as:

$$U(X_1 = x_1, ...X_n = x_n, a'_m) = \sum_{i=1}^{n} \theta_{(x_i,a'_m)} \tag{5.12}$$

   where $\theta_{(x_i,a'_m)}$ corresponds to the utility weight of the variable value x_i for the action a'_m. Note that the weights are specific to a given action value. The number of required parameters is here $|Val(a'_m)| \times (|Val(X_1)| + ...|Val(X_n)|)$. As a consequence, the size of the linear model is reduced to 581 independent parameters. This baseline hinges however on the assumption that the total utility of a given action can be decomposed as a linear combination of weights for each state variable value.

Rule-structured model

The two baselines were compared to a utility model structured with 15 utility rules. The interested reader is invited to browse the specification of these rules in Appendix B. The rule structure was designed by hand, while the parameters (in this case, utility values) remained unknown. The rules were associated to a total of 24 parameters.

Parameter estimation

Figure 5.8 offers a graphical comparison of the utility models produced for the two baselines and the rule-structured approach. The two baselines are essentially "flattened" or unstructured versions of the rule-based model. The input and output variables remain identical in all three models. However, the two baselines directly associate each state-action combination to a single utility value,

while the rule-structured approach defines this overall utility in a more indirect manner, through the instantiation of multiple utility rules.

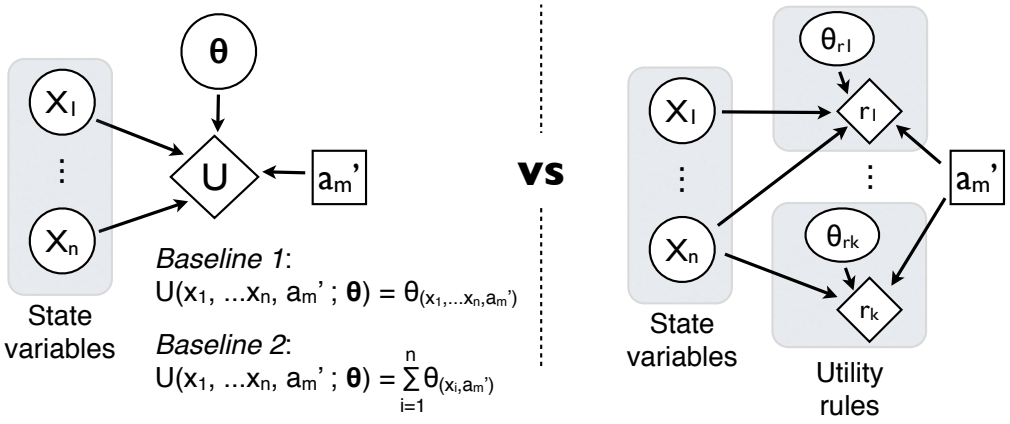


Figure 5.8: Baseline utility models (left) compared to the rule-structured utility model (right).

The parameter estimation procedure followed the Bayesian learning approach detailed in Section 5.2.2. For the baseline models, the function TRIGGERMODELS(...) resulted in the creation of one single utility node connected to the system action, as illustrated in Figure 5.8.

The parameter distributions for all utilities (both in the baseline and rule-structured models) were initialised with uniform priors on the interval $[-1, 6]$.

Evaluation

Given the utility models defined above, the action to execute corresponds to the action associated with the maximum utility in the current state. In this particular experiment, utility maximisation only considered the current (immediate) utility and did not perform forward planning.

The data collected from the Wizard-of-Oz interactions was split into a training set composed of 765 state-action pairs (75 % of the gathered data) and a held-out test set with 255 actions (remaining 25 %). The same training set was used to estimate the utility parameters for the three models. The resulting utility models were then evaluated on the basis of their *accuracy* – that is, the percentage of actions corresponding to the action selected by the wizard in the held-out test set. The accuracy results for the three models were evaluated at various stages of the estimation process in order to analyse and compare their learning performance. The accuracy figures were calculated by sampling over the parameters, performing inference over the resulting models, and finally averaging over the inference results.

5.3.4 Empirical results and analysis

Table 5.3 presents the accuracy results for the three utility models. The differences between the rule-structured model and the two baselines are statistically significant using Bonferroni-corrected paired t-tests, with p-value < 0.0001.

We performed an error analysis on the 17% of actions that deviate from the wizard behaviour.[6]

[6]One should be wary of labelling these actions as "incorrect", since they are in most cases relevant dialogue moves, but simply result from slightly different decision strategies than the one followed by the wizard.

The analysis revealed that the discrepancy is mainly due to two factors. The first factor is the lack of complete consistency on the part of the wizard, who occasionally decided to follow distinct strategies in similar situations (especially regarding the use of clarification requests). The second factor is the presence of a non negligible number of spurious and noisy data points, notably caused by e.g. interruptions in the middle of the experiments and technical issues with the control of the robotic platform (e.g. movements that had to be repeated due to motor failures).

Type of model	*Accuracy (in %)*
Plain model	67.35
Linear model	61.85
Rule-structured model	**82.82**

Table 5.3: Accuracy results for the three models on a held-out test.

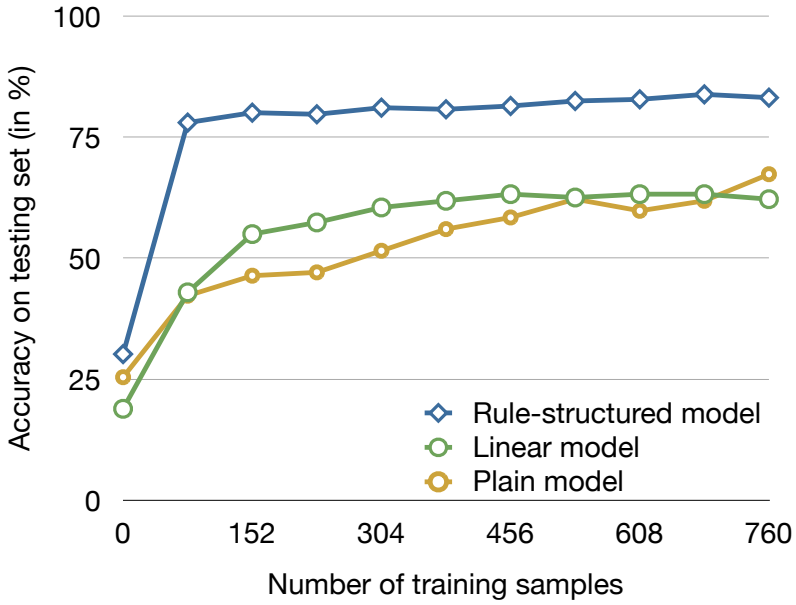
The learning curves for the three models are shown in Figure 5.9. Note that since the parameters are initially uniformly distributed, the accuracy is already non-zero before learning, since a random assignment of parameters has a low but non-zero chance of leading to the right action.

Thanks to its considerably reduced number of parameters, the rule-structured model is able to converge to near-optimal values after observing only a small fraction of the training set. The incorporation of domain knowledge via the rule structure has a clearly beneficial effect on the learning performance and on the generalisation capacity of the model. As the figure shows, the two baseline models do also improve their accuracies over time, but at a much slower rate. The linear model is comparatively faster than the plain model, but levels off towards the end. The suboptimal learning performance of the linear model is most likely due to the non-linearity of some dialogue strategies. The plain model continues its convergence and would probably reach an accuracy similar to the rule-structured model if given much larger amounts of training data.

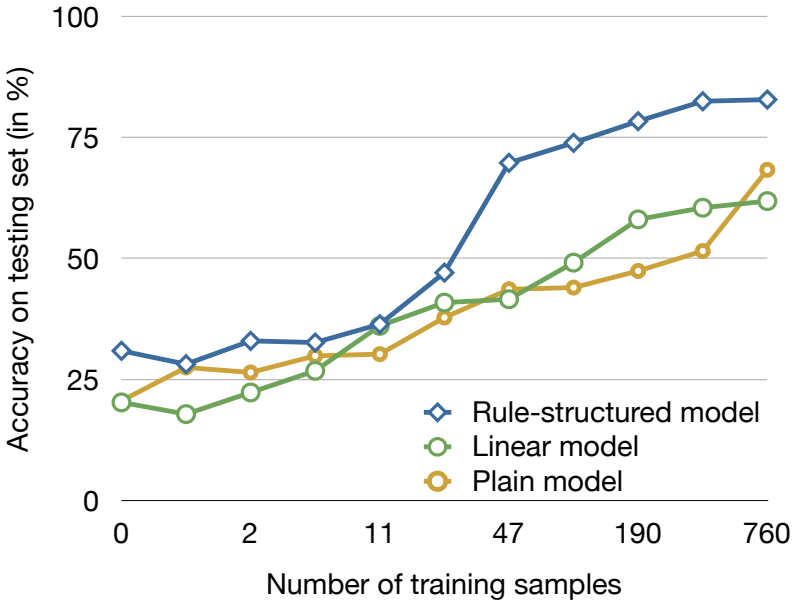
The learning results are in line with our expectations based on the respective sizes of the parameter space for the three utility models, and are not *per se* highly surprising. The main lesson to draw from this experiment is however not the exact difference in accuracy or learning rates for each particular model, but the fact that probabilistic rules can be successfully applied to structure a small but non-trivial dialogue domain and derive its parameters from collected interaction data. Through the use of abstraction mechanisms such as partitioning and quantification, the experiment demonstrates that the utility rules specified for the domain (see Appendix B) can cover large regions of the problem space without degrading the model performance. The utility model can hence be optimised with only modest amounts of in-domain data.

5.4 Conclusion

The present chapter described how parameters could be integrated in the specification of probabilistic rules and estimated via Bayesian learning techniques. Rule parameters can represent either probability or utility values. The estimation process operates by calculating posterior probability distributions over possible parameter values given the observed data set. The process is initiated with prior distributions such as Dirichlet priors for probability values and Gaussian or uniform priors for utilities.



(a) Linear scale



(b) Logarithmic scale (base 2)

Figure 5.9: Learning curves for the learned utility models on a held-out test set of 255 actions as a function of the number of processed data points. The accuracy results are given for the plain, linear and rule-structured utility models, using both linear (top) and logarithmic scales (bottom).

The main focus of the chapter was on supervised learning of rule parameters based on Wizard-of-Oz training data. We described how Wizard-of-Oz data can be practically collected and processed to yield data points encoded as pairs <dialogue state $\mathcal{B}_i$, wizard action a_i>. These data points are used to progressively narrow down the spread of the posterior distributions to the values that provide the best fit for the observed wizard actions.

This learning approach has been implemented in a spoken dialogue system for human-robot interaction and validated in a proof-of-concept experiment. The goal of the experiment was to estimate the utility values of various actions on the basis of a Wizard-of-Oz data set. Three utility models were compared: a plain utility table, a linear model, and a model structured with utility rules. The analysis of the empirical results shows that the rule-structured model outperforms the two baselines in regards to learning rate and generalisation performance.

The outcome of the experiment corroborates one of the central claims of this thesis – namely, that hybrid approaches to dialogue modelling (such as the probabilistic rules presented in this thesis) are well suited to model dialogue domains that must simultaneously confront high levels of uncertainty and limited amounts of in-domain training data. In such situation, which is commonplace in the field of spoken dialogue systems, neither purely symbolic nor purely data-driven approaches are alone sufficient to harness the complex and stochastic nature of the interactions. Hybrid approaches, on the other hand, provide ways to seamlessly combine expert knowledge and statistically optimised parameters in a single framework.

As shown in the experiments, Wizard-of-Oz interaction data can be a useful and interesting source of domain knowledge for the estimation of probabilistic models of dialogue. The data collection procedure can however be a tedious endeavour, as it requires:

1. The availability of an expert (the wizard) that can control the system and provide examples of appropriate behaviour for the domain.

2. The technical setup of a Wizard-of-Oz environment from which the wizard can perceive the user inputs, monitor contextual features, select possible actions to execute, and get all relevant information recorded and stored in a generic format.

A natural alternative to supervised learning from Wizard-of-Oz data is to let the dialogue system learn the best conversational behaviour via trial-and-error from its own interaction experience (that is, through reinforcement learning), without the reliance on external examples. The next chapter demonstrates how such strategy can be practically implemented with probabilistic rules.

Chapter 6

Learning from interactions

We extend in this chapter the parameter estimation approach to a reinforcement learning context. As explained in the first part of this thesis, a reinforcement learning agent learns how to act through a process of trial and error in a given environment. In our case, the environment is represented by verbal interactions with human users, and the system behaviour to learn corresponds to dialogue policies mapping dialogue states to relevant system responses.

The optimisation process is in many respects similar to the one outlined in the previous chapter. Bayesian inference remains the basic instrument for updating distributions over the model parameters on account of the collected evidence. However, the evidence is no longer represented by examples of expert behaviour as in supervised learning. The learning agent instead actively gathers experience through repeated interactions with (real or simulated) users, and receives feedback on its actions in the form of new observations and rewards. The parameter distributions are gradually refined on the basis of this feedback and subsequently used to select the next actions to execute. To minimise the number of parameters, the domain models are structured through probability and utility rules. This structured modelling approach allows the learning agent to escape the "curse of dimensionality" that often characterises dialogue domains. As a consequence, the number of interactions required to reach dialogue policies of high quality can be greatly reduced.

The chapter is divided in three sections. After a short survey of the core concepts of Bayesian reinforcement learning in Section 6.1, we expose in Section 6.2 our own reinforcement learning approach to the estimation of rule parameters. More specifically, we present how the parameters of probabilistic rules can be automatically optimised from interactions using either model-based or model-free strategies. Finally, Section 6.3 describes a practical experiment conducted in a human-robot interaction domain. The purpose of the experiment was to analyse the learning performance of rule-structured models compared to standard categorical distributions. Empirical results on a user simulator show that the rule-structured models converge to optimal parameters – and hence achieve higher rewards – in a much shorter time than unstructured representations.

6.1 Bayesian reinforcement learning

Bayesian reinforcement learning has recently emerged as a generic framework for learning and acting in uncertain environments (Poupart and Vlassis, 2008; Ross et al., 2011; Vlassis et al., 2012). As in other types of Bayesian learning methods, one core idea of Bayesian reinforcement learning is to maintain explicit probability distributions over the domain parameters and gradually narrow

down the spread of these distributions as more experience is collected. However, reinforcement learning agents are no longer mere passive observers in the interaction. The agent must indeed actively decide how to act after each turn in order to move the interaction forward.

As explained in Section 3.2, the actions selected by the agent must strike a balance between exploration (trying out new actions) and exploitation (preferring actions that are most likely to yield higher rewards). Bayesian reinforcement learning can offer principled solutions to the exploration-exploitation dilemma, as model uncertainty is explicitly accounted for in the action selection process (Duff, 2002; Ross et al., 2011). A Bayesian agent will therefore select actions that are expected to provide the highest long-term return given the current model uncertainty. When the uncertainty is high, information-gathering actions are preferred since they lead to a better understanding of the environment dynamics and are therefore more likely to result in higher future rewards. This inclination to explore gradually fades away as the learning agent develops a better grasp of its domain and becomes more confident about the relative merits of its own actions.

As for other families of reinforcement learning methods, Bayesian reinforcement learning can be classified in model-based and model-free methods.

Model-based methods

Model-based methods seek to learn an explicit model of the domain in the form of transition, observation and reward models. One benefit of model-based approaches is the relative simplicity of parameter estimation, as the model parameters can be directly updated upon the reception of each observation and reward using standard Bayesian inference. The policy is however complex to optimise, due to the combination of state uncertainty, stochastic action effects, and uncertainty over the parameters. The result is an augmented POMDP where the state also includes random variables expressing the model parameters in addition to traditional state variables. (Duff, 2002; Ross et al., 2011).

For domains of small to medium size, approximate dynamic programming methods can be applied to generate the α-vectors for this augmented POMDP. Point-based solvers (Pineau, 2004b; Porta et al., 2006) have notably shown reasonable performance on a variety of domains. These solution methods are however difficult to scale to more complex models due to the computational intractability of the optimisation process in the general case.

An alternative to offline approaches is online planning. Instead of compiling a policy for all possible states (as done in dynamic programming), online planning concentrates on selecting the best action for the current state at runtime. This selection is typically implemented in a search tree representing possible actions and their effects in terms of rewards and new observations. This tree is gradually expanded until a particular planning horizon is reached. Several approximate methods have recently been developed, based on e.g. point-based value iterations (Ross et al., 2008) and Monte Carlo tree search (Silver and Veness, 2010b). The action leading to the highest return on the search tree is then executed by the agent.

One non negligible benefit of online approaches is the possibility to dynamically adapt the domain models at runtime. This characteristic is important for dialogue domains where the domain models can vary in the course of the interaction – in order to e.g. adapt to shifting user preferences. Offline approaches must in comparison recalculate their policies after every modification or extension of the internal models for the domain. This advantage comes however at a price, namely

the fact that planning must be performed at execution time, while the interaction is taking place. Planning must therefore meet real-time constraints.

Interestingly, offline and online approaches are not mutually exclusive, but can be combined together to take full advantage of both strategies. The idea is to perform offline planning to pre-compute an initial rough policy, and use this policy as a heuristic approximation to guide the search of an online planner (Ross and Chaib-draa, 2007). These heuristic approximations can for instance be used to provide lower and upper bounds on the values associated with the dialogue states of the search tree. Based on these bounds, the planning algorithm can concentrate its computational efforts in the most fruitful regions of the search space and quickly discard irrelevant actions.

Model-based Bayesian reinforcement learning has been applied to dialogue management in several recent papers. Png et al. (2012) describe a generic Bayes-Adaptive POMDP framework and illustrate its use in simulated interactions. Doshi and Roy (2008b) present a similar POMDP framework with model uncertainty combined with active learning. Action selection is formalised in their paper by sampling possible POMDP models and extracting a solution for each sample. A related strategy is employed in a less principled manner by Atrash and Pineau (2009). Finally, Chinaei and Chaib-draa (2012); Chinaei et al. (2012) demonstrate the estimation of observation and reward models for dialogue POMDPs in an healthcare application. One interesting aspect of their work is the use of inverse reinforcement learning to automatically derive a reward model from expert policies.

Model-free methods

Model-free methods adopt a different learning strategy and directly optimise a dialogue policy from experience, without attempting to construct explicit internal models of the domain. One simple method, formalised by e.g. Dearden et al. (1998), is to assign prior distributions to the Q value estimates associated with state-action pairs, and iteratively refine these distributions upon the completion of each action. This update generally relies on Bellman's equation, since the Q values are never directly observed (only the observations and rewards are available to the agent). The optimal action is then simply defined as the one that maximises the Q values for the current state, modulo an "exploration bonus" added at learning time to favour exploratory strategies.

Other Bayesian model-free approaches rely on Gaussian processes, which extend the above approach to problems with continuous state and actions spaces (Engel et al., 2005), and policy gradient methods, which directly optimise a parametrised policy by gradient ascent (Baxter and Bartlett, 2001; Ghavamzadeh and Engel, 2006).

Gašić et al. (2011) present a framework for dialogue policy optimisation based on Gaussian processes. One of the main benefits of their approach is the tremendous acceleration of the optimisation procedure. As a result, the dialogue policy can be optimised via live interactions with human users instead of being confined to simulation. Daubigney et al. (2012a) describe a related approach based on Kalman Temporal Differences. As their approach is grounded in Kalman filtering instead of full Bayesian filtering, it only estimates the first and second moment of the parameter distributions – i.e. its mean and variance – instead of the full posterior distribution (Geist and Pietquin, 2010). Finally, Bayesian learning approaches have also been applied to partially observable dialogue domains which necessitate the estimation of a transition model to update the dialogue state, even though the dialogue policy itself is optimised in a model-free manner (Thomson et al., 2010).

Scalability

Bayesian reinforcement learning has been the subject of much recent research in the last decade, based on both model-based and model-free paradigms. This research focus led to the development of powerful optimisation methods (see e.g. Vlassis et al., 2012, for a detailed survey). Scalability remains nevertheless a important concern when porting these methods to real applications. The sizes of the parameter and action spaces are in particular major bottlenecks for many learning methods, especially in partially observable domains.

As argued in the next section, probabilistic rules can contribute to addressing by reducing the number of parameters and filtering out irrelevant actions from the planning process.

6.2 Optimisation of rule parameters

We developed in this thesis two distinct approaches to the optimisation of rule parameters from unannotated interactions. Both employ Bayesian reinforcement learning as theoretical framework and probabilistic rules as representation formalism, but follow distinct optimisation strategies:

- The first approach follows a model-based strategy. In this approach, the rule-structured models $\mathcal{M}$ of the domain correspond to transition, observation and reward models. The models are associated to a collection of parameters $\boldsymbol{\theta}$ with prior distributions $P(\boldsymbol{\theta})$. These parameter distributions are updated on the basis of the observations and rewards received by the system during the interactions. Forward planning is used at runtime to calculate the expected cumulative utilities of possible actions and select the one yielding the maximum utility given the current dialogue state and rule parameters.

- The second approach is a model-free strategy. The transition and reward models are here replaced by a collection of parametrised utility rules representing the estimated Q value for the system actions. In contrast to the model-based strategy, the utility parameters are here updated via a temporal-difference learning method. The actions to execute are determined through an ϵ-greedy policy that strikes a balance between the selection of known high-utility actions and the exploration of new actions.

The two sections below flesh out these two approaches in more detail.

6.2.1 Model-based approach

The model-based approach relies on the specification of probabilistic rules that describe:

- the *transition model* for the domain, i.e. how the dialogue state is likely to change as a result of the system actions

- the *observation model* for the domain, i.e. what are the likely observations associated with a given dialogue state;

- the *reward model* for the domain, i.e. what are the immediate utilities (reflecting the system objectives) that result from the execution of particular system actions.

Figure 6.1 depicts a dynamic decision network where transition model is encoded in the figure as a probability distribution $P(s_t \mid s_{t-1}, a_{t-1}\,; \boldsymbol{\theta}_T)$, the observation model by a probability distribution $P(o_t \mid s_t\,; \boldsymbol{\theta}_O)$ and the reward model by the utility distribution $R_t(s_t, a_t\,; \boldsymbol{\theta}_R)$. For the sake of clarity, the figure abstracts away from the rule nodes mediating between the variables in the network, and upon which the parameters are attached.

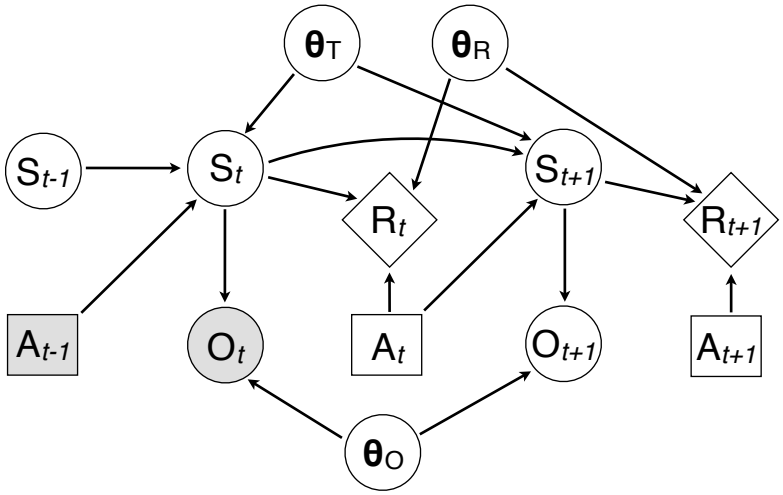


Figure 6.1: Dynamic decision network for a model-based learning strategy.

Parameter estimation

The probabilistic rules corresponding to these three models may all include unknown parameters that must be estimated from data. Fortunately, this worst case scenario where $\boldsymbol{\theta}_T$, $\boldsymbol{\theta}_R$ and $\boldsymbol{\theta}_O$ must all be learned from scratch rarely happens in practice. As exemplified in the experiment described at the end of this chapter, the reward and observation models can often be defined by the system designer prior to learning.

Two types of information sources are available to the agent to refine its parameters during the interaction: the observations and the rewards. In the model-based setting, parameter update is relatively straightforward. The key idea is to include the parameter variables as part of the dialogue state. The probability distributions of these parameters are then automatically updated as part of the state update process (see Algorithm 7 in Section 4.4). There is therefore no need for special purpose mechanisms beyond standard Bayesian update.

Example of parameter update

Let us illustrate this process on the domain example from Section 4.4.3. The example of dialogue snippet remains unchanged:

USER : Now move forward
$\tilde{a}_u = \langle (Request(Forward), 0.6), (Request(Backward)), 0.4) \rangle$

SYSTEM : Could you please repeat?

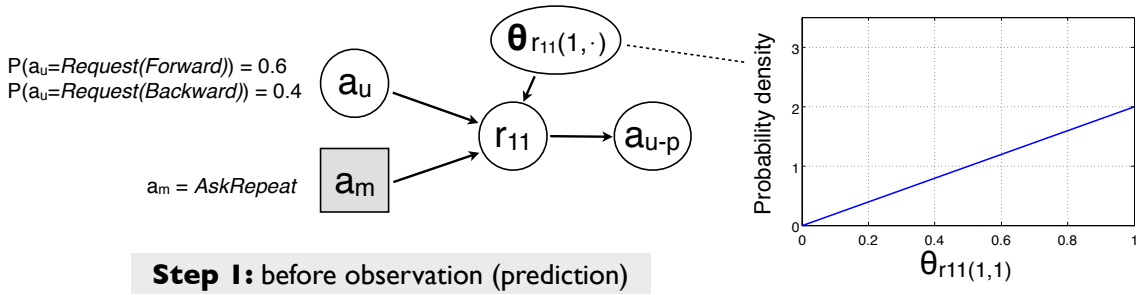
USER : Please move forward!

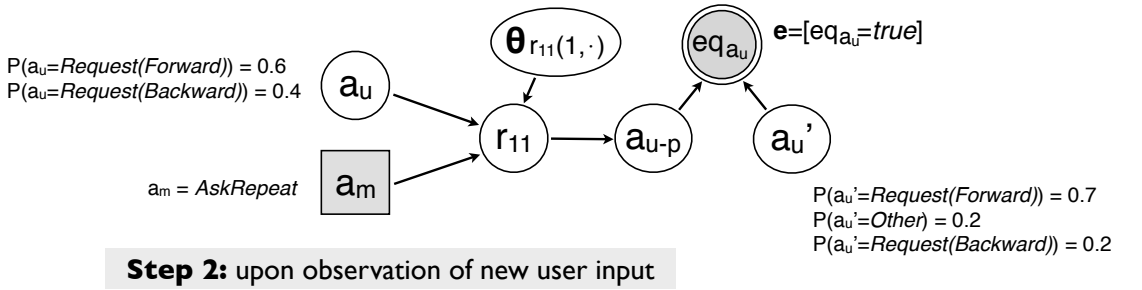
$$\tilde{a}_u = \langle (Request(Forward), 0.7), (Other, 0.2), (Request(Backward), 0.1) \rangle$$

We now assume that the effect distribution associated with the predictive rule r_{11} is unknown and replaced by parameters:

$$r_{11} : \forall y : \\ \textbf{if } (a_m = AskClarify \wedge a_u = y) \textbf{ then} \\ \begin{cases} P(a_{u\text{-}p} = y) = \theta_{r_{11}(1,1)} \\ P([\cdot]) = \theta_{r_{11}(1,2)} \end{cases}$$

Figure 6.2 illustrates how the distribution over the parameter values for $\boldsymbol{\theta}_{r_{11}(1,\cdot)}$ is automatically modified as part of the state update operation. To keep the procedure as simple as possible, the figure ignores the steps related to action selection and concentrates on the application of rule r_{11}. The prior distribution $P(\boldsymbol{\theta}_{r_{11}(1,\cdot)})$ is set in this example to $\sim \text{Dirichlet}(2, 1)$, as we can reasonably presuppose that the user is more likely than not to repeat her/his last utterance after an explicit request from the system.





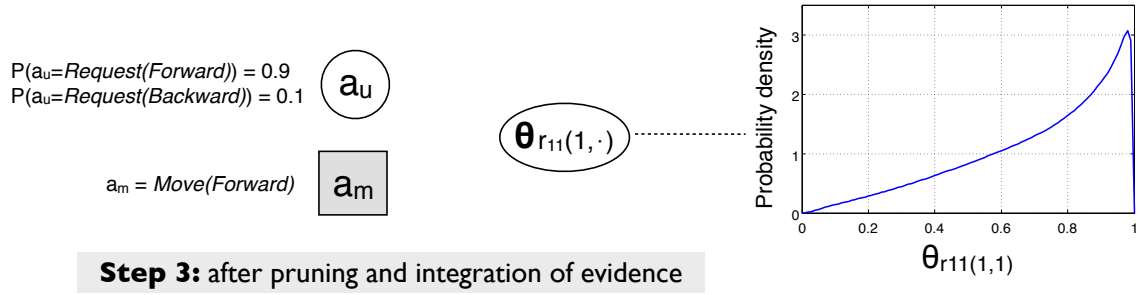


Figure 6.2: Parameter update for $\boldsymbol{\theta}_{r_{11}(1,\cdot)}$ after the reception of a new user input.

The first step illustrates the instantiation of the rule along with its parameter. Upon the reception of a new observation in the form of a user input a'_u, the dialogue state and parameters are updated (step 2). After pruning and integration of the evidence (step 3), we notice that the parameter distribution $P(\boldsymbol{\theta}_{r_{11}(1,\cdot)})$ has shifted part of its probability mass further to the right. In other words, the probability of the user repeating his last utterance becomes somewhat more likely. The posterior distribution $P(\boldsymbol{\theta}_{r_{11}(1,\cdot)})$ after the update is constructed using kernel density estimation.

Action selection

As the Q values are not directly accessible in model-based approaches, action selection must resort to either dynamic programming or forward planning to calculate the expected future rewards of each action. Action selection is the computational bottleneck in model-based Bayesian reinforcement learning, since the agent needs to reason not only over all the current and future states, but also over all possible models (parametrised by the $\boldsymbol{\theta}$ variables). The high dimensionality of the task often prevents the use of offline solution techniques. We apply in this work a simple forward planning algorithm coupled with importance sampling.

Algorithm 13 selects the action to execute through forward planning on a given horizon h. The selection procedure relies on the recursive function CALCULATE-Q-VALUES($\mathcal{B}, \mathbf{e}, h$) to compute the Q-values of possible actions given a current state $\mathcal{B}$, evidence $\mathbf{e}$ and planning horizon h.

Algorithm 13 : PLANACTION ($\mathcal{B}$, e, h)

Input: Dialogue state $\mathcal{B}$ as a decision network
Input: Evidence $\mathbf{e}$
Input: Planning horizon h
Output: Selected action $\mathbf{a}^*$

1: $Q_{\mathcal{B}} \leftarrow$ CALCULATE-Q-VALUES ($\mathcal{B}, \mathbf{e}, h$)
2: Find optimal value $\mathbf{a}^* = \text{argmax}_{\mathbf{a}}\, Q_{\mathcal{B}}(\mathbf{a})$
3: Remove utility nodes from the state $\mathcal{B}$
4: **return** $\mathbf{a}^*$

Algorithm 14 : CALCULATE-Q-VALUES ($\mathcal{B}, \mathbf{e}, h$)

1: Let $\mathbf{A}$ be the set of all decision variables in $\mathcal{B}$
2: **for all** possible action $\mathbf{a} \in Val(\mathbf{A})$ **do**
3: $\quad Q_{\mathcal{B}}(\mathbf{a}) \leftarrow U_{\mathcal{B}}(\mathbf{a}, \mathbf{e})$
4: $\quad$ **if** $h > 1$ **then**
5: $\quad\quad \mathcal{B}' \leftarrow$ dialogue state updated from $\mathcal{B}$ after action $\mathbf{a}$
6: $\quad\quad$ **for all** possible observation $\mathbf{o}$ **do**
7: $\quad\quad\quad \mathcal{B} \leftarrow$ dialogue state updated from $\mathcal{B}'$ after observation o
8: $\quad\quad\quad Q_{\mathcal{B}''} \leftarrow$ CALCULATE-Q-VALUES($\mathcal{B}'', \mathbf{e}, h-1$)
9: $\quad\quad\quad Q_{\mathcal{B}}(\mathbf{a}) \leftarrow Q_{\mathcal{B}}(\mathbf{a}) + \gamma\, P_{\mathcal{B}'}(\mathbf{o})\ \max_{\mathbf{a}'} Q_{\mathcal{B}''}(\mathbf{a}')$
10: $\quad\quad$ **end for**
11: $\quad$ **end if**
12: **end for**
13: **return** $Q_{\mathcal{B}}$

The Q value of an action is the discounted addition of its immediate reward (line 3 in Algorithm 14) and the expected future reward following its execution (line 4-11). Line 6 loops on possible observations. For efficiency reasons, only a limited number of high-probability observations are selected. For each observation, the dialogue state is updated (line 7) and the Q values for the resulting state $\mathcal{B}''$ are computed (line 8). The maximum Q value for this future state is then added to the Q value for the current state, weighted by the discount factor γ and the probability $P_{\mathcal{B}'}(\mathbf{o})$ of the observation. The procedure stops when the planning horizon has been reached, or the algorithm has run out of time. The planner then simply selects the action with maximum expected cumulative utility.

Algorithm 14 contains two loops: one cycling over the set of possible actions, and one cycling over the set of possible observations following the system action. This process can be represented in an AND-OR search tree anchored in the current state. The OR branches denote the system actions along with their respective rewards, while the AND branches denote the observations weighted by their likelihood. An example of such AND-OR search tree is provided in Figure 6.1, which is modified from Ross et al. (2008).

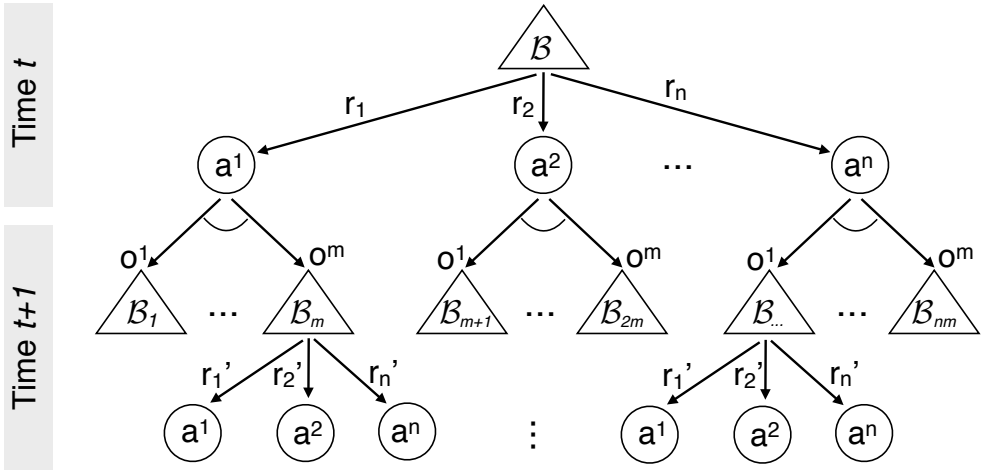


Figure 6.3: AND-OR search tree constructed through forward planning for a horizon of 2.

Our practical implementation of Algorithms 13 and 14 operates in anytime mode and expands the search tree by gradually adding new observations and actions in a breadth-first manner. At any point in time, the planning algorithm is thus able to deliver a solution. The quality of the solution will of course depend on the accuracy of the search tree, which itself depends on the number of sampled actions and observations – more trajectories leading to a more accurate plan, but at a higher computational cost. The anytime nature of the algorithm is important since the planner operates online and must thus satisfy real-time constraints.

As argued in Lison (2012b), the reliance on utility rules to encode the reward model can help making the planning process more tractable. In addition to assigning utilities to system actions, utility rules also implicitly define the set of action values that are relevant at a given time.[1] In other words, actions that are not deemed relevant in a particular state are automatically filtered out from the planning process. Instead of searching through the whole space of possible actions, the

[1] Recall that in Algorithm 5 from Section 4.3, the set of possible values of an action node is directly derived from the values listed in the utility nodes connected to it.

planning algorithm is thus limited a subset of actions that are locally relevant. One interesting aspect of this approach is that the filtering of relevant actions is done solely on the basis of the provided reward model and does not require the integration of additional constraints or ad-hoc mechanisms. This stands in contrast with most existing work in dialogue management, where constraints on possible actions are typically enforced through external filtering techniques defined on the basis of e.g. information-state update rules (Heeman, 2007), finite-state automata (Williams, 2008b), or – in our own previous work on this problem – high-level constraints encoded in Markov Logic formulae (Lison, 2010b).

Learning cycle

The model-based learning cycle is detailed in Algorithm 15. The learning agent incrementally updates its model parameters $\boldsymbol{\theta}$ by running a number of interactions, either with a real user or in simulation. Starting with an initial dialogue state, the interaction alternates between the reception of new observations (in the form of e.g. user inputs or contextual changes in the environment) and the execution of system actions following these observations. The dialogue state is updated after each observation (line 5), using the procedure outlined in Section 4.4, with the action selection method replaced by Algorithm 13. The selected actions are then executed (line 7). If the reward model is unknown, the reward resulting from the system actions can be integrated in the set of observations and used to update the reward parameters accordingly.

Algorithm 15 : MODEL-BASED-RL-LEARNING $(\mathcal{M}, \mathcal{B}_0, \boldsymbol{\theta}, N)$

Input: Rule-structured models $\mathcal{M}$ for the domain
Input: Initial dialogue state $\mathcal{B}_0$
Input: Model parameters $\boldsymbol{\theta}$ with prior distribution $P(\boldsymbol{\theta})$
Input: Number N of interactions to collect
Output: Posterior distribution $P(\boldsymbol{\theta})$ for the parameters

1: **for** $i = 0 \rightarrow N$ **do**
2: Start new interaction with initial state $\mathcal{B} = \mathcal{B}_0 \cup \boldsymbol{\theta}$
3: **while** interaction is active **do**
4: Get new observations $\mathbf{O}$
5: UPDATESTATE($\mathcal{B}, \mathbf{O}$)
6: **if** non-empty selected action a in $\mathcal{B}$ **then**
7: Execute action a and get resulting reward r
8: **end if**
9: **end while**
10: **end for**
11: **return** $P(\boldsymbol{\theta})$

6.2.2 Model-free approach

In parallel to the model-based learning strategy described above, we also developed an alternative Bayesian model-free approach to the estimation of rule parameters. In this approach, the core model that is to be estimated is the *action-value model* which specifies the expected cumulative reward Q of the system actions depending on the current state.

In addition to this action-value model, the domain may also include transition and observation models. However, the transition and observation models are in model-free approaches only used for state update (if the dialogue state contains hidden state variables that are indirectly inferred from observations, such as the user intentions) and are not exploited in the action selection. This stands in contrast with model-based approaches to reinforcement learning where transition and observation models are directly employed to plan the next action.

Figure 6.4 illustrates how these distributions combine to form a dynamic decision network. As for the model-based approach, all domain models are specified with probabilistic rules (which are again abstracted away from the simplified diagram in the figure). The transition and observation models are encoded by probability rules and the action-value model by utility rules.

In the worst case, all models may include unknown parameters. The transition model is thus defined as a probability distribution $P(s_t \mid s_{t\text{-}1}, a_{t\text{-}1} ; \boldsymbol{\theta}_T)$, the observation model by a distribution $P(o_t \mid s_t ; \boldsymbol{\theta}_O)$ and the Q value model by a distribution $Q_t(s_t, a_t ; \boldsymbol{\theta}_Q)$.

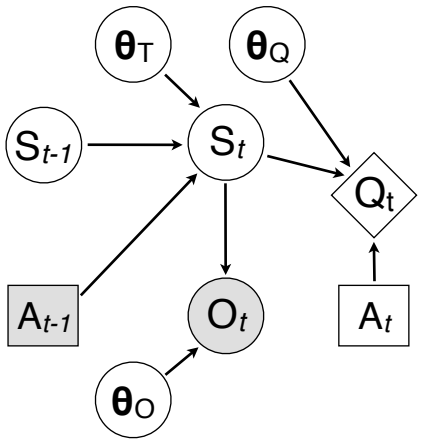


Figure 6.4: Dynamic decision network for a model-free strategy.

Parameter estimation

The transition and observation models can be estimated in the same manner as in the model-based approach – that is, by including the parameters in the dialogue state and refining their distributions as part of the state update process.

The estimation of the action-value model is however slightly more intricate, since the Q values are not directly accessible to the learning agent. The only feedback perceived by the agent are indeed the immediate rewards resulting from its actions and the subsequent observations, not the expected cumulative rewards Q. A solution to this estimation problem is to rely on Bellman's equation to incrementally improve the action-value estimates on the basis of the rewards resulting from the agent actions. Such methods are called temporal-difference methods and includes many popular reinforcement learning algorithms such as SARSA and Q-learning (Sutton and Barto, 1998). Temporal-difference methods are also called "boostrapping" methods as they approximate new Q value estimates based on previously learned estimates.

The particular model-free learning method applied in this work is based on the well-known SARSA algorithm.[2] The classical, MDP-based definition of SARSA proceeds as follows. Let s_t be a dialogue state at time t, followed by a system action a_t. The execution of the system action a_t results in a reward r_{t+1} and a new dialogue state s_{t+1} which is itself followed by a second system action a_{t+1}. The SARSA update of the Q value estimate for the first action a_t is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{6.1}$$

where α represents the learning rate of the algorithm. The estimate $Q(s_t, a_t)$ is thus modified in direction of the value $[r_{t+1} + \gamma \, Q(s_{t+1}, a_{t+1}]$, with a learning step expressed by α.

[2] SARSA stands for State-Action-Reward-State-Action, which is a reference to the processing order of the algorithm.

The approach developed in this thesis rests on a simple Bayesian extension of SARSA. The posterior distribution over the parameters is here computed on the basis of the evidence provided by the reward r_{t+1} and next system action a_{t+1}. Given a sequence of state-action-rewards $\langle \mathcal{B}_t, a_t, r_{t+1}, \mathcal{B}_{t+1}, a_{t+1} \rangle$, we define the likelihood distribution $P(r_{t+1}, a_{t+1} \,;\boldsymbol{\theta})$ as:

$$P(r_{t+1}, a_{t+1} \,;\boldsymbol{\theta}) = \phi\left(\alpha\left[Q_{\mathcal{B}_t}\left(a_t \,;\boldsymbol{\theta}\right) - \left(r_{t+1} + \gamma\, Q_{\mathcal{B}_{t+1}}\left(a_{t+1} \,;\boldsymbol{\theta}\right)\right)\right]\right) \tag{6.2}$$

where $\phi(\cdot)$ is the density function for the standard normal distribution $\mathcal{N}(0,1)$. The standard normal distribution has its peak around the value 0 and decreases exponentially with the distance to this mean. The likelihood distribution will therefore yield a high probability when the initial estimate $Q_{\mathcal{B}_t}(a_t \,;\boldsymbol{\theta})$ available at time t is close to the updated estimate $\left[r_{t+1} + \gamma\, Q_{\mathcal{B}_{t+1}}(a_{t+1} \,;\boldsymbol{\theta})\right]$ at time $t+1$, and a low probability otherwise. The learning rate α controls the relative weight of this distance between estimates.

Based on the likelihood distribution defined in Equation (6.2), the posterior distribution on the parameters is finally rewritten as:

$$P(\boldsymbol{\theta} \,|\, r_{t+1}, a_{t+1}) = \eta\, P(r_{t+1}, a_{t+1} \,;\boldsymbol{\theta})\; P(\boldsymbol{\theta}) \tag{6.3}$$

Action selection

The above section described how the parameter distributions were updated on the basis of the received rewards and executed actions, but did not explain how the system actions were selected at runtime. A simple strategy is to select the action yielding the maximum Q value for the current state. This greedy strategy can however result in poor control policies whenever the agent gets stuck in a suboptimal behaviour. Greedy strategies can be improved by allowing the agent to explore other actions once in a while. The relative frequency of these exploration actions compared to the "greedy" actions is expressed by the probability ϵ, which is usually small. This method is called an ϵ-greedy strategy and is illustrated in Algorithm 16.

Algorithm 16 : ϵ-GREEDY-POLICY $(\mathcal{B}, \mathbf{e})$

Input: Dialogue state $\mathcal{B}$ as a decision network
Input: Evidence $\mathbf{e}$
Output: Selected action $\mathbf{a}^*$

1: Select value $\mathbf{a}^* = \begin{cases} \operatorname{argmax}_{\mathbf{a}} Q(\mathbf{a}, \mathbf{e}) & \text{with probability } (1-\epsilon) \\ \text{another action} & \text{with probability } \epsilon \end{cases}$
2: Remove utility nodes from the state $\mathcal{B}$
3: **return** $\mathbf{a}^*$

Learning cycle

The model-free learning cycle is mostly similar to the one defined in the model-based setting. As shown in Algorithm 17, the agent estimates the values of its rule parameters by collecting a number of interactions. Each interaction starts from the initial dialogue state $\mathcal{B}_0$ and unfolds as a sequence of observations and actions. The dialogue state contains both traditional state variables and parameter variables. After perceiving new observations, the dialogue state is correspondingly

updated – including transition and observation parameters if present in the domain (line 5). The update process comprises the selection of new system actions, according to the ϵ-greedy selection procedure shown in Algorithm 16. When a new action is selected, the posterior distributions over parameters are correspondingly updated through temporal-difference learning (line 7). The action is then executed and the resulting reward is retrieved (line 8). The process is repeated for each interaction.

Algorithm 17 MODEL-FREE-RL-LEARNING $(\mathcal{M}, \mathcal{B}_0, \boldsymbol{\theta}, N)$

Input: Rule-structured models $\mathcal{M}$ for the domain
Input: Initial dialogue state $\mathcal{B}_0$
Input: Model parameters $\boldsymbol{\theta}$ with prior distribution $P(\boldsymbol{\theta})$
Input: Number N of interactions to collect
Output: Posterior distribution $P(\boldsymbol{\theta})$ for the parameters

1: **for** $i = 0 \rightarrow N$ **do**
2: Start new interaction with initial state $\mathcal{B} = \mathcal{B}_0 \cup \boldsymbol{\theta}$
3: **while** interaction is active **do**
4: Get new observations $\mathbf{O}$
5: UPDATESTATE($\mathcal{B}$, $\mathbf{O}$)
6: **if** non-empty selected action a in $\mathcal{B}$ **then**
7: Update posterior $P(\boldsymbol{\theta} \mid r, a)$ based on Equation (6.3)
8: Execute action a and get resulting reward r
9: **end if**
10: **end while**
11: **end for**
12: **return** $P(\boldsymbol{\theta})$

6.3 Experiments

We performed an empirical evaluation of the two outlined approaches to Bayesian reinforcement learning based on a user simulator for a human-robot interaction domain. The evaluation is divided in two parts:

1. The goal of the first experiment was to determine whether the use of probabilistic rules could be shown to improve the performance of a reinforcement learning agent. More specifically, the experiment compared two alternative formalisations of a transition model for a human-robot interaction scenario: one encoded with traditional categorical distributions, and one encoded with probability rules.

2. The goal of the second experiment focused on the comparison between model-based and model-free approaches in Bayesian reinforcement learning. The evaluation compared a model-based learner with an unknown transition model (such as the one used in the first experiment) to a model-free learner with an unknown action-value model. Both strategies relied on probabilistic rules to capture their respective models and were evaluated on the basis of their average rewards when interacting with the user simulator.

We first describe in this section the dialogue domain and user simulator used in both experiments, and then detail the evaluation setups and empirical results for each experiment.

6.3.1 Dialogue domain

As in the previous chapter, the dialogue domain chosen for the experiments is a human-robot interaction scenario with a Nao robot. The task is however more complex than the one developed for the supervised learning case. The interactions collected for the experiments involved the Nao robot conversing with a human user in a shared visual scene including a few graspable visual objects, as illustrated in Figure 6.5. The users were instructed to command the robot to carry the objects from one place to another. The users were free to decide which object(s) to pick up, where to place them on the floor, and what kinds of navigation commands to provide to perform the task. In addition to following the human instructions, the robot could also answer factual questions from the user regarding its own knowledge of the environment such as "*do you see a blue cylinder?*" or "*what do you see?*".

Each (physical or information-gathering) sub-task is represented as a distinct user intention. As the human users could only instruct the robot to perform one sub-task at a time, the user intention is represented by a single variable denoting the current sub-task that the user wish to see fulfilled. The user intentions for the domain are listed in Table 6.1.

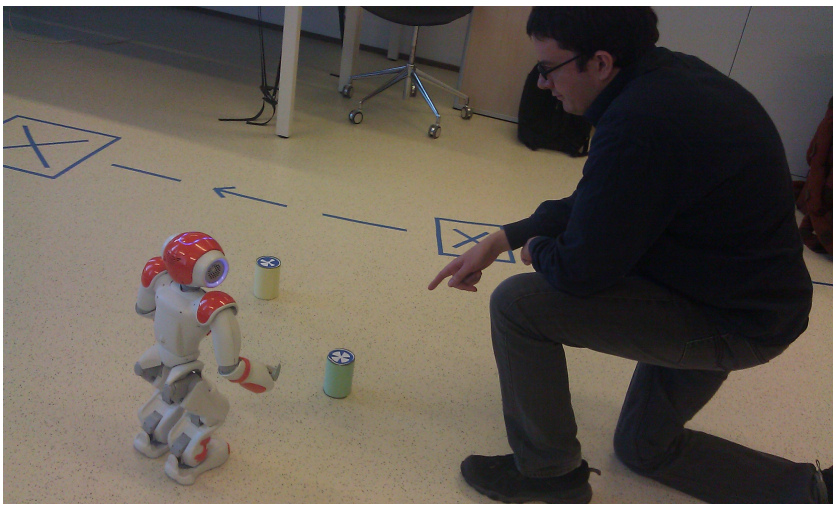


Figure 6.5: Human user interacting with the Nao robot in a shared visual scene with two objects.

The objects in the scene consisted of coloured metallic cylinders with a special marking on their top to facilitate the robot's visual servoing during grasping tasks. As the Nao robot used in the experiments did not include actuated fingers, the grasping operation employed permanent magnets attached to the robot hands to grasp and carry the cylinders.

In addition to following the user commands related to spatial navigation and object manipulation, the robot could also perform grounding-related actions such clarification requests and acknowledgements. In total, the domain included 9 user dialogue templates. The robot has a repertoire of 8 possible action templates that can be executed. For a dialogue domain with two objects, the user actions a_u and system actions a_m will thus expand into respectively 15 and 37 actions. Tables 6.2 and 6.3 list the possible actions respectively available for the user and system.

· Move(x)
  where $x = \{\text{Left}, \text{Right}, \text{Forward}, \text{Backward}\}$
· PickUp(x)
  where x is an object identifier
· Release(x)
  where x is an object identifier
· WhatDoYouSee
· DoYouSee(x)
  where x is an object identifier

Table 6.1: List of user intentions i_u

· Ask(Move(x))
  where $x = \{\text{Left}, \text{Right}, \text{Forward}, \text{Backward}\}$
· Ask(PickUp(x))
  where x is an object identifier
· Ask(Release(x))
  where x is an object identifier
· RepeatLastMove
· Ask(WhatDoYouSee)
· Ask(DoYouSee(x))
  where x is an object identifier
· Confirm
· Disconfirm
· Other

Table 6.2: List of user actions a_u

· Do(x)
  where $x = \{\text{Move}(y), \text{PickUp}(z), \text{Release}(z)\}$
  and $y = \{\text{Left}, \text{Right}, \text{Forward}, \text{Backward}\}$
  and $z =$ an object identifier
· Excuse(x)
  where $x = \{\text{DoNotSeeObject}, \text{DoNotCarryObject}, \text{AlreadyCarryObject}\}$
· Describe(x)
  where $x =$ a (possibly empty) list of object identifiers
· ConfirmDetection
· DisconfirmDetection
· Ground(x)
  where $x = \{\text{Move}(y), \text{PickUp}(z), \text{Release}(z)\}$
  and $y = \{\text{Left}, \text{Right}, \text{Forward}, \text{Backward}\}$
  and $z =$ an object identifier
· AskClarify
· AskConfirm(x)
  where $x = \{\text{Move}(y), \text{PickUp}(z), \text{Release}(z), \text{DoYouSee}(z), \text{WhatDoYouSee}\}$
  and $y = \{\text{Left}, \text{Right}, \text{Forward}, \text{Backward}\}$
  and $z =$ an object identifier

Table 6.3: List of system actions a_m

Transcript 3 provides a detailed example of recorded interaction between a human user and the (wizard-controlled) robot.

The reward model was defined by hand, using standard schemes: the execution of correct actions or the correct answer to user questions leads to large positive values (+6 in this particular case) while the execution of wrong or irrelevant actions leads to large negative values (-6) and the use of clarification or confirmation requests to small negative values (from -0.5 to -1.5, depending on the type of request). Table 6.4 presents the reward model defined for the domain.

Action	**Reward**
Execution of correct physical action $a_m = \text{Do}(i_u)$	+6
Execution of wrong physical action $a_m = \text{Do}(x)$ with $x \neq i_u$	-6
Declare $a_m = \text{Excuse}(\text{DoNotSeeObject})$ when $i_u = \text{PickUp}(x)$ and x is not perceived	+6
Declare $a_m = \text{Excuse}(\text{DoNotCarryObject})$ when $i_u = \text{Release}(x)$ and x is not carried	+6
Declare $a_m = \text{Excuse}(\text{AlreadyCarryObject})$ when $i_u = \text{PickUp}(x)$ and x is carried	+6
Declare $a_m = \text{Excuse}(*)$ in other circumstances	-6
Correct answer $a_m = \text{Describe}(x)$ when $i_u = \text{WhatDoYouSee}$ and x are the perceived objects	+6
Wrong answer $a_m = \text{Describe}(*)$ when $i_u \neq \text{WhatDoYouSee}$	-6
Correct answer $a_m = \text{ConfirmDetection}$ when $i_u = \text{DoYouSee(x)}$ and x is perceived	+6
Wrong answer $a_m = \text{ConfirmDetection}$ when $i_u \neq \text{DoYouSee(x)}$ or x is not perceived	-6
Correct answer $a_m = \text{DisconfirmDetection}$ when $i_u = \text{DoYouSee(x)}$ and x is not perceived	+6
Wrong answer $a_m = \text{DisconfirmDetection}$ when $i_u \neq \text{DoYouSee(x)}$ or x is perceived	-6
Grounding of correct intention $a_m = \text{Ground}(i_u)$	+2
Grounding of wrong intention $a_m = \text{Ground}(x)$ with $x \neq i_u$	-6
Request to confirm correct intention $a_m = \text{AskConfirm}(i_u)$	-0.5
Request to confirm wrong intention $a_m = \text{AskConfirm}(x)$ with $x \neq i_u$	-1.5
Request to clarify $a_m = \text{AskClarify}$	-1
Ignore user act $a_m = \text{None}$ when $a_u \neq \text{None}$	-1.5

Table 6.4: Reward model for the domain.

6.3.2 Simulator

Generalities

In order to draw meaningful and reliable comparisons between reinforcement learning approaches, the user behaviours must be made fully consistent across interactions. This consistency must be enforced on both the conversational choices of the user and the average amount of noise and comprehension errors that characterise them. Needless to say, this criteria is hard to satisfy when working with human participants. The comparative evaluation of learning approaches was thus conducted with the help of a simulator.

The simulator is responsible for two tasks:

1. The first task is to emulate the typical dialogue behaviour of an human user, and generate relevant user responses to the system actions.

Transcript 3 User interaction with wizard-controlled robot

ROBOT: Hello human

HUMAN: Hi, hello Lenny!

[ROBOT *stands up*]

HUMAN: Would you please move forward

ROBOT: OK, moving forward

[ROBOT *steps forward*]

HUMAN: Would you please move a bit more forward

ROBOT: Should I move backward

HUMAN: No, a bit more forward

ROBOT: OK, forward then

[ROBOT *steps forward*]

HUMAN: Do you see any object?

ROBOT: Sorry I do not understand

HUMAN: Do you see any object?

ROBOT: I see a blue cylinder

HUMAN: Can you pick it up please?

ROBOT: Should I pick up the object?

HUMAN: Yes please

ROBOT: OK, thanks

[ROBOT *moves in direction of the blue cylinder*]
[ROBOT *picks up the object*]

ROBOT: Done

HUMAN: Now turn right

[ROBOT *turns right while holding the object*]

HUMAN: Move forward

[ROBOT *moves forward while holding the object*]

HUMAN: And then move two steps backward

ROBOT: OK, moving forward

[ROBOT *starts movement*]

HUMAN: No please move backward

[ROBOT *stops*]

[...]

2. The second task is to maintain a virtual representation of the environment during the interaction (represented in our case by the physical objects) and update this representation as a function of the system actions

It should however be emphasised that the reliance on user simulators to conduct the comparative evaluation of our learning approaches does not in any way imply that simulators are a necessary component of the learning approach presented in this thesis.[3]

Wizard-of-Oz study

In order to build a user simulator that matches as closely as possible the behaviour of actual human subjects, we started by recording a set of Wizard-of-Oz interactions in the human-robot dialogue domain chosen for the experiments.

The technical setup employed for the Wizard-of-Oz data collection was mostly similar to the one described in the previous chapter. The purpose of the data collection are however not identical: while the Wizard-of-Oz interactions described in Section 5.3.2 served to determine the most appropriate *system* actions depending on the situation, the goal of the Wizard-of-Oz interactions for the present experiments is to collect empirical data about the most likely *user* actions in their context. As the wizard behaviour was not the focus of the study, the wizard was here allowed to directly listen to the user utterances, without using the speech recogniser as intermediary. The wizard controlled the verbal and physical actions via a remote screen coupled to the robotic platform. Various types of errors and misunderstandings were artificially introduced by the wizard in the course of the interaction in order to collect data about the user responses to such comprehension errors.

A total of eight interactions were recorded, each with a different speaker, totalling about 50 minutes divided in 486 turns. The interactions were performed in English. The users were again recruited amongst the local group of students and employees in the Department of Informatics at the University of Oslo and were (with one exception) non-native English speakers. The author of the present thesis served as the wizard.

After the recording, the dialogues were segmented and annotated by hand. The first layer of annotation encodes the user dialogue acts a_u and system actions a_m, according to the lists in Table 6.2 and 6.3. The user intentions i_u underlying the user commands are annotated on top of this sequence of turns. [4] The annotation also includes two contextual variables respectively expressing the lists of objects perceived and carried by the robot at a given time. Transcript 4 provides a concrete example of annotation for the first part of the interaction in Transcript 3.

User and situation modelling

After collecting and annotating the Wizard-of-Oz interactions, the next step in the development of the simulator is to design the (stochastic) transition model that determines how the user and the environment are to respond to the system actions. This statistical model is used at runtime to sample possible responses and feed them back to the dialogue system.

[3] In fact, probabilistic rules are particularly well suited for optimisation from live interactions, as they require drastically less training data than traditional learning approaches (as evidence by the results in this chapter).

[4] Although the user intentions are in principle hidden "mentalistic" entities, they can in our domain be easily determined by a human annotator from the dialogue transcript.

Transcript 4 Annotated dialogue excerpt

HUMAN: Would you please move forward

Annotation: $a_u = \text{Ask}(\text{Move}(\text{Forward}))$
$i_u = \text{Move}(\text{Forward}), \textit{carried} = [], \textit{perceived} = []$

ROBOT: OK, moving forward

Annotation: $a_m = \text{Ground}(\text{Move}(\text{Forward}))$ *and* $\text{Do}(\text{Move}(\text{Forward}))$

HUMAN: Would you please move a bit more forward

Annotation: $a_u = \text{Ask}(\text{Move}(\text{Forward}))$
$i_u = \text{Move}(\text{Forward}), \textit{carried} = [], \textit{perceived} = [\textit{object}_1]$

ROBOT: Should I move backward

Annotation: $a_m = \text{AskConfirm}(\text{Move}(\text{Backward}))$

HUMAN: No, a bit more forward

Annotation: $a_u = \text{Disconfirm}$ *and* $\text{Ask}(\text{Move}(\text{Forward}))$,
$i_u = \text{Move}(\text{Forward}), \textit{carried} = [], \textit{perceived} = [\textit{object}_1]$

ROBOT: OK, forward then

Annotation: $a_m = \text{Ground}(\text{Move}(\text{Forward}))$ *and* $\text{Do}(\text{Move}(\text{Forward}))$

HUMAN: Do you see any object?

Annotation: $a_u = \text{Ask}(\text{WhatDoYouSee})$,
$i_u = \text{WhatDoYouSee}, \textit{carried} = [], \textit{perceived} = [\textit{object}_1]$

ROBOT: Sorry I do not understand

Annotation: $a_m = \text{AskClarify}$

HUMAN: Do you see any object?

Annotation: $a_u = \text{Ask}(\text{WhatDoYouSee})$,
$i_u = \text{WhatDoYouSee}, \textit{carried} = [], \textit{perceived} = [\textit{object}_1]$

ROBOT: I see a blue cylinder

Annotation: $a_m = \text{Describe}([\textit{object}_1])$

The transition model for our human-robot interaction dialogue domain is factored in four components:

1. a user goal model $P(i'_u \mid i_u, a_m, \textit{perceived}', \textit{carried}')$ describing the probability of the next user intention i'_u as a function of the current intention i_u, the last system action a_m and the two contextual variables *perceived* and *carried*.[5]

2. a user action model $P(a'_u \mid i'_u, a_m)$ describing the probability of the new user action a'_u as a function of the new user intention and last system action.

3. a contextual model $P(\textit{perceived}' \mid \textit{perceived}, a_m)$ describing how the set of objects perceived by the robot evolves as a function of the system action (since a movement may result in the

[5]The user intention can depend from the contextual variables since the user is e.g. more likely to ask the robot to grasp an object if it sees it, and less likely to ask the same request if the robot already carries an object.

detection of new objects or make other objects fall from view).

4. a contextual model $P(carried' \mid carried, a_m)$ describing how the set of objects carried by the robot can change due to grasping and releasing actions.

The resulting probabilistic model that combines these four distributions is depicted in Figure 6.6. The reader may notice that the current state variables at time t are greyed out, signifying that their values are observed. It should be stressed that the knowledge of these values is limited to the simulator (since the user is aware of its own intentions and dialogue acts, and the environment "knows" its own state). The robot has however no access to the internal state of the simulator.

The transition model was practically designed with a set of probability rules expressing the four distributions based on a small number of structural assumptions about the user behaviour. The probabilities associated with the rule effects were then estimated by maximum likelihood on the basis of the annotated dialogues.

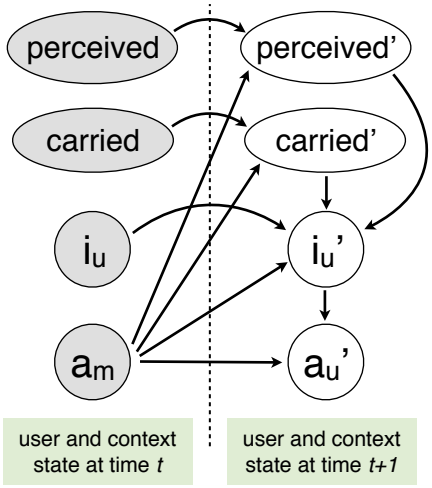


Figure 6.6: User and context models employed by the simulator.

Error modelling

In real interactions, user inputs are not directly observed by the dialogue manager but must first be processed by the speech recognition and understanding modules. These modules are prone to various failures and frequently distort or misinterpret the actual user utterance. The user simulator should account for this fact by explicitly modelling errors and uncertainties arising from speech recognition and natural language understanding.

In order to reproduce the imperfect nature of the communication channel, the simulator wraps every user input in a N-best list of the following form:

$$\tilde{a}_u = \begin{cases} P(\text{correct } a_u) = p_1 \\ P(\text{another randomly selected value for } a_u) = p_2 \\ P(\text{spurious recognition}) = p_3 \end{cases}$$

where $\langle p_1, p_2, p_3 \rangle$ are probability values sampled at runtime from a three-dimensional Dirichlet distribution. The distribution $P(p_1, p_2, p_3)$ is estimated in an empirical manner based on actual speech recognition results for the domain. We first applied the off-the-shelf speech recogniser embarked on the robot (Nuance Vocon) to the audio segments corresponding to the user utterances collected in the Wizard-of-Oz study. The recognition results were then processed in order to extract from each audio segment the three probabilities $\langle p_1, p_2, p_3 \rangle$, where p_1 stands for the probability of the correct utterance (which can be zero if the utterance does not appear in the N-Best list), p_2 the total probability of incorrect utterances, and p_3 the probability of no recognition. We finally derived a Dirichlet distribution based on these sample probabilities using the estimation method developed by T. Minka (Minka, 2003). The particular Dirichlet distribution resulting from the recognition results was $\sim \mathsf{Dirichlet}(5.4, 0.52, 1.6)$

At runtime, the probabilities for the N-best list elements are drawn from the Dirichlet distribution. Probability values falling below a minimum threshold are automatically pruned from the N-Best list. The method has been found to match reasonably well the actual recognition results produced by the speech recogniser, although one could naturally refine the approach by e.g. explicitly modelling confusion probabilities between individual inputs.

Simulation procedure

The simulation procedure takes the form of an interaction loop between the simulator and the dialogue system, as shown in Figure 6.7. The simulation comprises two dialogue architectures interacting with one another: the simulator on the one hand and the control system for the robot on the other hand. Note that the two systems operate differently, as the simulator has a fully observable state, while the system state is only partially observable. Another obvious difference is the fact that the domain models of the system contain unknown parameters to optimise, while the simulator models are known and fixed.

Contrary to the experiments presented in the previous chapter, the dialogue architecture used in this learning experiment is essentially reduced to the dialogue management module, since the simulator and the dialogue system directly exchange their dialogue actions without needing to express them in actual spoken utterances.

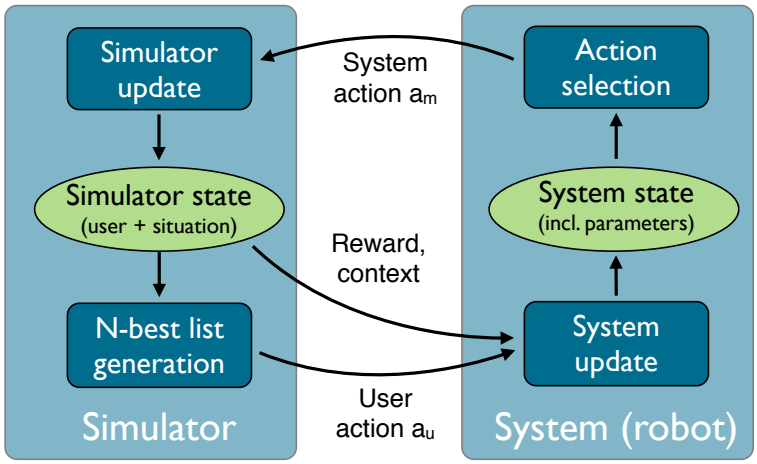


Figure 6.7: Processing workflow for the simulated interaction.

An example of interaction generated when coupling the simulator to the dialogue system is provided in Transcript 5. The excerpt is taken at the onset of the learning cycle, when the system's knowledge of the domain models is still rudimentary, which explains the prevalence of information-gathering actions.

6.3.3 First experiment

The goal of the first experiment is to determine whether the use of probabilistic rules has a beneficial influence on the learning performance of the agent. The motivation is sensibly the same as the one

Transcript 5 Example of simulated interaction

[*Initial context:* perceived = [$\text{object}_1, \text{object}_2$], carried = []]

HUMAN: $\begin{bmatrix} \text{Ask}(\text{PickUp}(\text{object}_2)) & 0.06 \\ \text{Ask}(\text{PickUp}(\text{object}_1)) & 0.42 \end{bmatrix}$

ROBOT: AskRepeat $\rightarrow$ Reward = -1

HUMAN: $\begin{bmatrix} \text{Ask}(\text{DoYouSee}(\text{object}_2)) & 0.01 \\ \text{Ask}(\text{PickUp}(\text{object}_1)) & 0.77 \end{bmatrix}$

ROBOT: $\text{Ground}(\text{PickUp}(\text{object}_1))$ $\rightarrow$ Reward = 2

ROBOT: $\text{Do}(\text{PickUp}(\text{object}_1))$ $\rightarrow$ Reward = 6, carried = [object_1]

HUMAN: $\begin{bmatrix} \text{Ask}(\text{DoYouSee}(\text{object}_2)) & 0.05 \end{bmatrix}$

ROBOT: AskRepeat $\rightarrow$ Reward = -1

HUMAN: $\begin{bmatrix} \text{Ask}(\text{DoYouSee}(\text{object}_2)) & 0.87 \end{bmatrix}$

ROBOT: $\text{AskConfirm}(\text{DoYouSee}(\text{object}_2))$ $\rightarrow$ Reward = -0.5

HUMAN: $\begin{bmatrix} \text{RepeatLast} & 0.05 \\ \text{Confirm} & 0.63 \end{bmatrix}$

ROBOT: ConfirmDetection $\rightarrow$ Reward = 6

HUMAN: $\begin{bmatrix} \text{Ask}(\text{Move}(\text{Forward})) & 0.73 \\ \text{Ask}(\text{PickUp}(\text{object}_2)) & 0.03 \end{bmatrix}$

ROBOT: AskRepeat $\rightarrow$ Reward = -1

HUMAN: $\begin{bmatrix} \text{Ask}(\text{Move}(\text{Forward})) & 0.92 \\ \text{Ask}(\text{Move}(\text{Left})) & 0.04 \end{bmatrix}$

ROBOT: $\text{Ground}(\text{Move}(\text{Forward}))$ $\rightarrow$ Reward = 2

ROBOT: $\text{Do}(\text{Move}(\text{Forward}))$ $\rightarrow$ Reward = 6

[...]

put forward in the experiment of the previous chapter, except the estimation procedure is here based on model-based Bayesian reinforcement learning techniques instead of supervised learning.

The experiment focuses more precisely on the estimation of the transition model for the human-robot interaction domain described in Section 6.3.1. Based on the simulator presented in the previous pages, the experiment compares two alternative representations of the transition model: one baseline model encoded via standard categorical distributions, and one equivalent model encoded via probability rules. The relative performance of these two representations is measured by the average return – i.e. the sum of rewards – per interaction.

The reward model was held fixed and identical in both cases (cf. Table 6.4). Online planning was used for action selection and operated with a horizon of length 2.

Baseline model

The transition model $P(s' \mid s, a_m)$ is represented in the baseline approach through traditional factored categorical distributions. The transition model is more precisely divided into a user goal model $P(i'_u \mid i_u, a_m)$ and a user action model $P(a'_u \mid i'_u, a_m)$.[6] The user goal model is defined in the following manner:

$$P(i'_u \mid i_u, a_m) = \begin{cases} P(i'_u) & \text{if } a_m \text{ fullfils the intention } i_u \\ 1 & \text{if above condition does not hold and } i'_u = i_u \\ 0 & \text{otherwise} \end{cases}$$

where $P(i'_u)$ is a categorical distribution that expresses the prior probability of a new user intention i'_u. The user action model $P(a'_u \mid i'_u, a_m)$ is for its part constructed as a plain probability table where each possible assignment of values for the parent variables i'_u and a_m is assigned a distinct categorical distribution on the values of a'_u.

The resulting parameters for these categorical distributions are encoded with Dirichlet priors. The baseline model used for this experiment contains a total of 228 Dirichlet parameters. Weakly informative priors are used to initialise the prior distributions.

Rule-structured model

The rule-structured model is encoded with parametrised probability rules. A total of six rules with 13 corresponding Dirichlet parameters (of varying dimensions) is used to define the transition model. As for the baseline model, the rule parameters are initially associated with weakly informative Dirichlet priors. The rules designed for the experiment are listed in Appendix B.

Empirical results

The performance was first measured in terms of average return per simulated interaction, shown in Figure 6.8. To analyse the accuracy of the transition model, we also derived the Kullback-Leibler divergence (Kullback and Leibler, 1951) between the next user act distribution $P(a'_u)$ predicted by the learned model and the actual distribution followed by the simulator at a given time, as shown in

[6] The two contextual variables perceived and carried are not included in the transition model since their values are fully observable and do not need to be predicted in advance.

Figure 6.9. Some residual discrepancy is to be expected between these two distributions, the latter being based on the actual user intention while the former must infer it from the current belief state. The results of both figures are averaged on 100 simulations.

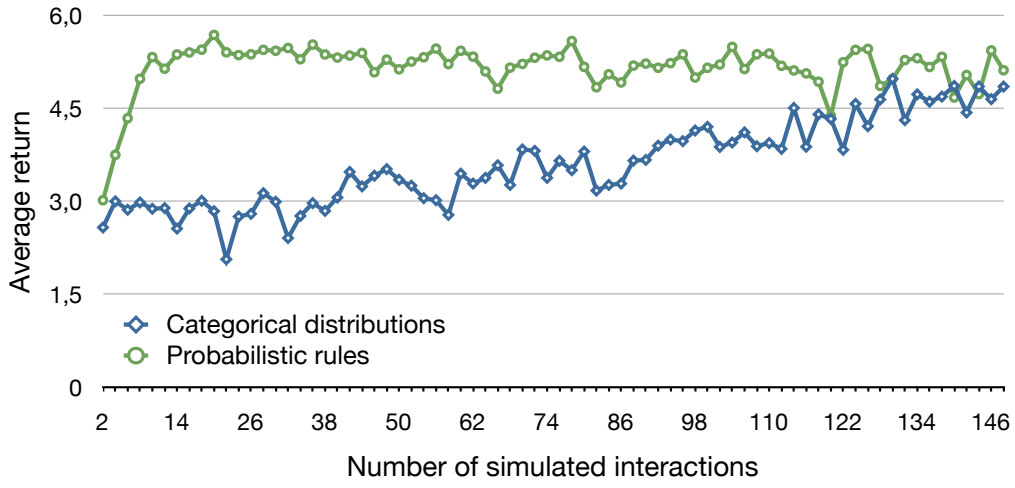


Figure 6.8: Average return as a function of the number of simulated interactions.

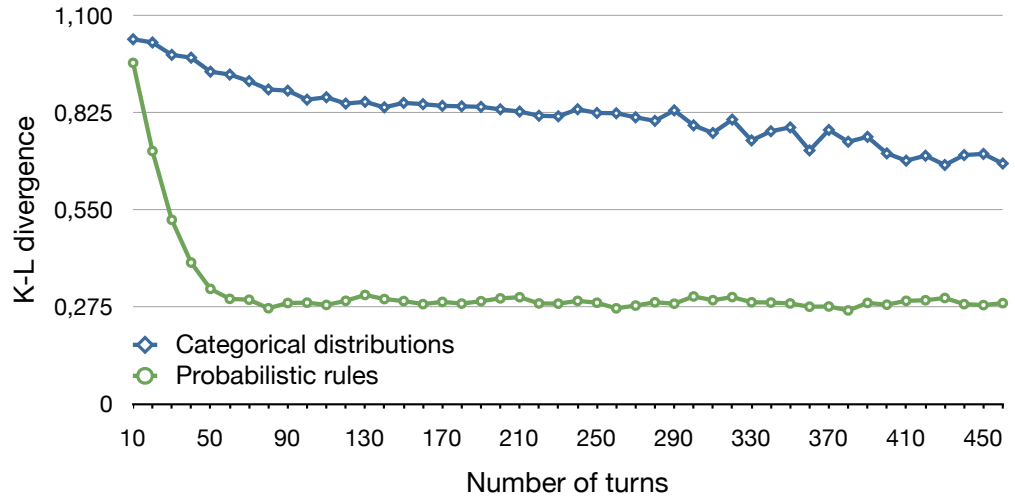


Figure 6.9: Kullback-Leibler divergence between the distribution $P(a'_u)$ estimated from the current model and the actual distribution followed by the simulator.

Analysis of results

The empirical results illustrate that both models are able to capture at least some of the interaction dynamics and achieve higher returns as the number of turns increases, but they do so at different learning rates. In our view, this difference is to be explained by the higher generalisation capacity of the probabilistic rules compared to the unstructured categorical distributions.

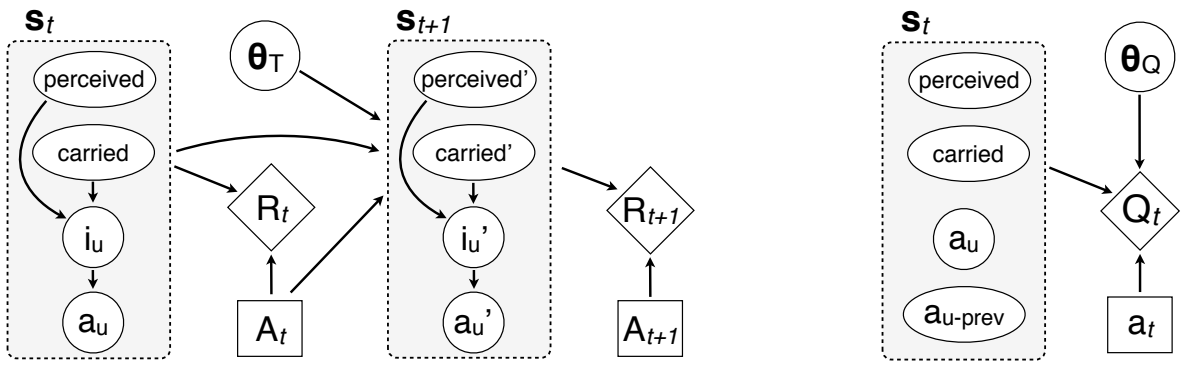
One can observe from the empirical results that the Dirichlet parameters associated with the probabilistic rules converge to their optimal value very rapidly, after a handful of episodes. This is a promising result, since it implies that the proposed approach could in principle optimise dialogue policies from live interactions, without resorting to a user simulator.

One caveat is nevertheless here in order. As described in the previous section, the simulation model used to sample the next intentions and actions of the user is built on a number of structural assumptions. The learning results must therefore be interpreted with caution, as real-world dialogues may not exhibit the same kind of internal structure as the one derived from the simulator. We shall however see in Chapter 8 that the results presented here fortunately also carry over to genuine human-robot interactions.

6.3.4 Second experiment

The second experiment focused on the comparison of model-based and model-free approaches to the estimation of rule parameters via reinforcement learning. As in the previous experiment, the reward model is provided but the transition model is unknown. Both approaches are encoded in this experiment with probabilistic rules.

In the model-free case, the rule parameters encode the action-value model $Q(s, a)$ over the return of state-action pairs, while the model-based case focuses on the transition model $P(s' \mid a, s)$. Figure 6.10 illustrates these two learning strategies.



(a) Model-based approach (b) Model-free approach

Figure 6.10: Model-based (left) and model-free (right) approaches compared in the experiment.

Model-based approach

The model-based approach is identical to the one presented in the previous experiment. The transition model is thus structured with six rules associated with 13 corresponding Dirichlet parameters of varying dimensions. The Dirichlet parameters are again initialised with weakly informative Dirichlet priors. As for the first experiment, the action selection was performed with an online planner operating with a horizon of 2.

Model-free approach

The model-free approach relied on a utility model structured with 12 probabilistic rules associated with 27 Gaussian parameters. As no transition model is here available to infer the underlying user intentions, the dialogue state is defined on the basis of the recent history of dialogue acts (last user action a_u, last system action a_m and preceding action $a_{u\text{-}prev}$) as well as the list of objects currently perceived and carried by the robot.

The resulting rule parameters were optimised using the SARSA-based method outlined in Section 6.2.2.

Empirical results

The simulator was coupled to the dialogue system to compare the learning performance of the two methods. Figure 6.11 shows the average returns over 100 iterations.

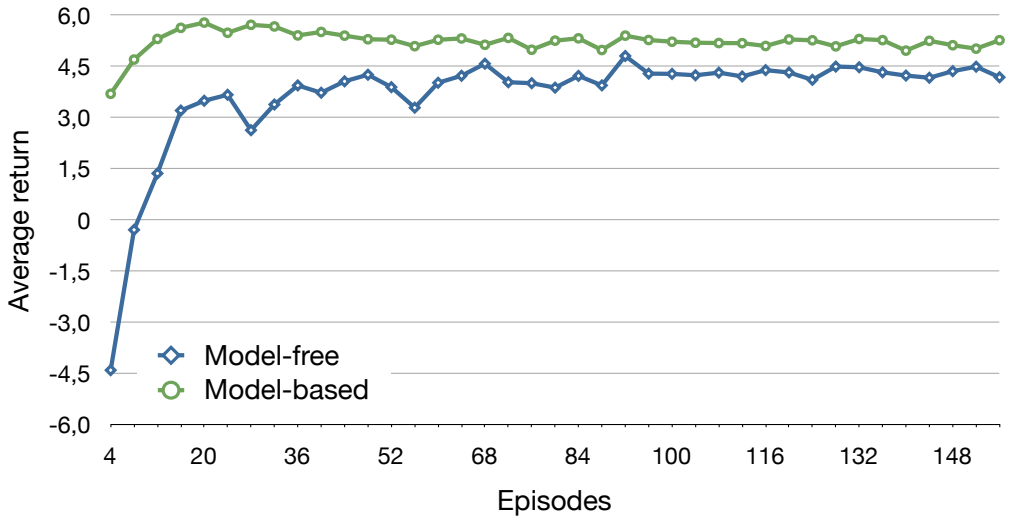


Figure 6.11: Average return as a function of the number of episodes

Analysis of results

We can notice on Figure 6.11 that the model-based approach converges to a high-quality policy in fewer episodes than its model-free counterpart. This comes however at the cost of higher computational demands brought about by the need to perform online planning, as evidenced in Figure 6.12 by the roughly similar times required for convergence (around 50 min.). We can also observe that the model-based approach yields slightly higher returns in this experiment, although this difference is harder to explain. One hypothesis is that the model-based approach can accumulate evidence about the underlying user intention in its belief state, while its model-free equivalent lacks an explicit transition model and can therefore only ground its decisions on the history of dialogue acts and objects perceived by the robot.

The results suggest that the model-based approach outperforms its competitor in this type of learning contexts. This conclusion is however subject to some caveats. First, the tractability of the model-based approach is directly dependent on the length of the planning horizon. Domains with

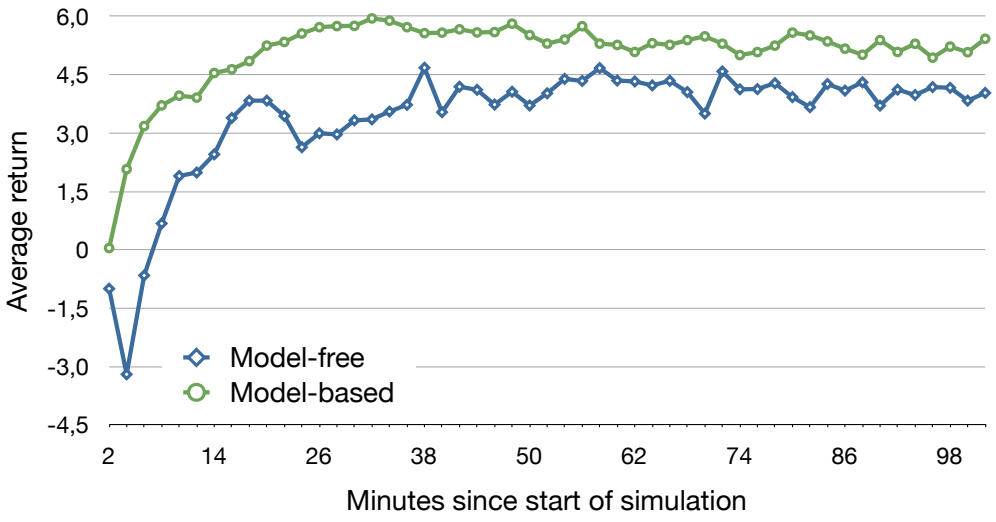


Figure 6.12: Average return as a function of the elapsed time

longer horizons might be better addressed with a model-free strategy due to the exponential growth of the search tree. Second, the model-free learning algorithm presented here remains relatively simple – as it is based on a Bayesian variant of the well-known SARSA algorithm – and more advanced frameworks such as Gaussian Processes (Gašić et al., 2011) might yield different model-free results. Finally, it should be noted that the model-based method could rely in this work on the availability of the reward model. This is not an unreasonable assumption for dialogue domains, as the reward model is often a reflection of the application objectives as specified by the system designer. It may nevertheless be difficult to specify all aspects of the reward model at design time. Whether the results carry out to cases where the reward model must also be (partially or fully) estimated remains an open question.

The model-free approach presented in this experiment only focused on the estimation of the action-value model and did not include a transition model. An interesting idea for future work would be to blend the two optimisation methods and simultaneously estimate a transition model together with the action-value model. Such hybrid approach would combine the advantages of model-free and model-based strategies, since it would allow the action-value model to be grounded in high-level variables such as the user intentions instead of remaining confined to observation variables. Action selection could be performed in such hybrid framework via a mixture of offline and online planning as in Ross and Chaib-draa (2007).

6.4 Conclusion

We exposed in this chapter two alternative Bayesian reinforcement learning techniques to the optimisation of parameters associated with probabilistic rules.

The first technique is a model-based approach that explicitly constructs statistical models of the domain in the form of transition, observation and reward models. The range of possible values for the domain parameters are represented as prior probability distributions that are incrementally updated based on the observations and rewards perceived during the interactions. Action selection is then performed through forward planning on the basis of the current dialogue state and domain

parameters. Probabilistic rules are used to structure the domain model and thereby reduce the number of parameters to optimise.

The second technique is a model-free approach that skips the estimation of the domain models to directly construct an action-value model expressing the Q values of state-action pairs. As the Q values are never directly observed, the update of Q values is bootstrapped based on Bellman's equation. We described in particular how the action-value model could be structured with utility rules and optimised through a Bayesian variant of the SARSA learning algorithm combined with ϵ-greedy action selection.

The last section detailed two experiments conducted in a simulated environment to evaluate the empirical performance of the two learning approaches. The simulator was common to both experiments and constructed on the basis of a Wizard-of-Oz study conducted in a human-robot interaction domain. The simulator included three distinct models, all estimated from the collected Wizard-of-Oz data: (1) a *user model* expressing how the user intentions and dialogue acts are likely to evolve as a function of the system actions, (2) a *context model* expressing the objects in the visual field of the robot as well as the ones carried by the robot, and (3) an *error model* introducing errors and inaccuracies into the generated user input in order to mimic the imperfect nature of speech recognition

On the basis of this user simulator, the first experiment compared the performance of model-based Bayesian reinforcement learning on two alternative formalisations of the transition model. The first formalisation relied on traditional categorical distributions, while the second represented the transition model through probability rules. The empirical results showed that the rule-structured model could converge to a high-performing dialogue behaviour – as measured by the average return per interaction – much faster than the baseline.

The second experiment examined the relative performance of model-based and model-free approaches. The two compared models were in this experiment structured with probabilistic rules. The results showed a slightly better performance of the model-based approach. We however noted that this difference was contingent on specific aspects of the experimental setup such as the length of the planning horizon.

The previous and present chapter demonstrated how probabilistic rules can facilitate the optimisation of dialogue policies both in the supervised and reinforcement learning case. The experiments have nevertheless so far concentrated on the *learning* performance of rule-structured approaches, and have not yet evaluated the effects of probabilistic rules on interaction success and user satisfaction. Chapter 8 will address this important question. But before doing so, we first discuss the technical implementation of the data structures and algorithms exposed in this thesis. This is the subject of the next chapter.

Chapter 7

Implementation

CHAPTER NOT READY YET!!

This chapter exposes the most important features of the openDial toolkit, which is a Java-based software toolkit developed to design and evaluate dialogue systems based on probabilistic rules. The toolkit implements all the data structures and algorithms detailed in this thesis. It also served as a experimental platform to carry out the empirical studies presented in Chapters 5, 6 and 8.

The chapter is divided in three sections. The first section summaries the most important characteristics of the dialogue architecture and surveys the system modules integrated in the system and the graphical user interface developed to monitor and control the system state. Section 7.2 focuses on the declarative specification of the probabilistic rules for the domain. Section 7.3 then discusses a range of technical questions related to the implementation of algorithms related to approximate inference, sampling, and forward planning. Finally, Section 7.4 compares openDial to other architectures developed in dialogue systems research.

7.1 Architecture

7.1.1 General workflow and scheduling

he dialogue system developed for this thesis relies on a blackboard, event-driven architecture. As already mentioned throughout this thesis, blackboard architectures are widely used in spoken dialogue systems for their ability to handle flexible workflows where multiple modules "cooperate" to interpret the user inputs, maintain a representation of the current situation, and decide on the best actions to perform. Information-state approaches are notably based on such system architecture Larsson and Traum (2000b). In dialogue domains, the blackboard represents the dialogue state. The system modules are in charge of updating this dialogue state in accordance with the recognised user utterances, perceived contextual changes, and selected system actions.

Figure 7.1 provides a high-level illustration of the workflow corresponding to such architecture. One can easily note from the figure that the dialogue state stands at the center of the workflow. The system modules can both read (dashed arrows) and write (plain arrows) to this dialogue state. The scheduling of these processing operations is done in an event-driven manner. After each change, the dialogue state sends an event message to all its attached modules to inform them that the state has been updated. The modules can subsequently react to this update by generating new updates. The process continues until the dialogue state stabilises.

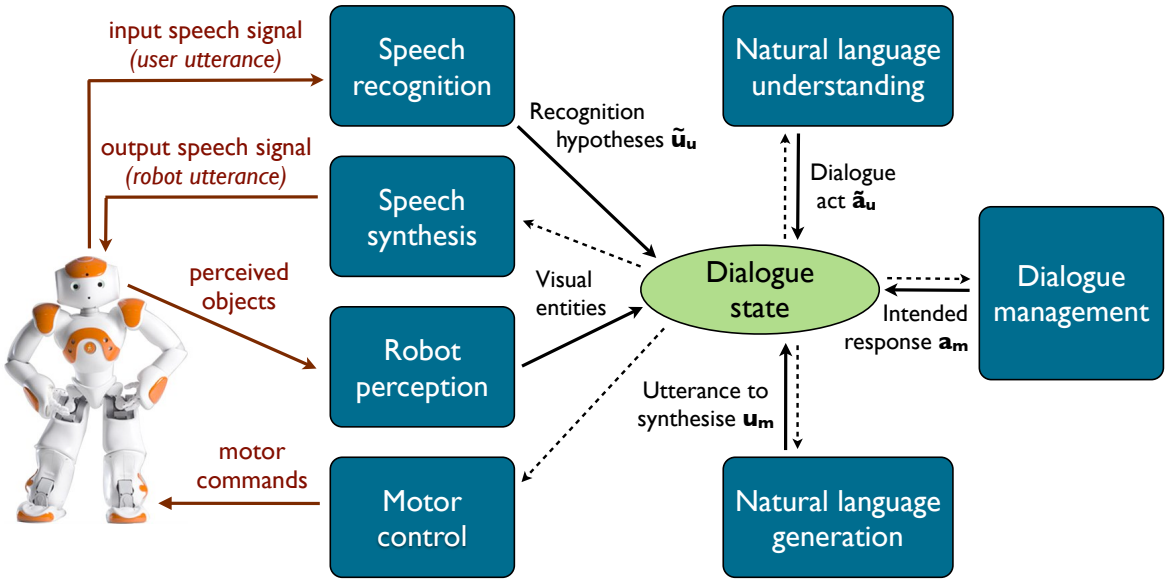


Figure 7.1: Generic system architecture employed in this thesis.

The openDial toolkit allows system modules to run in parallel, using Java multi-threading. The possibility to execute modules in parallel is particularly useful in dialogue architectures, as the system must be able to react to user input and contextual changes occurring at any time – even while other modules are still busy processing a previous update. Many of the modules developed in the toolkit (such as planning and probabilistic inference) can also operate in anytime mode, which implies that they can be interrupted at any time and still provide an output.

The current implementation of openDial is optimised to run on a single platform. System modules can however remotely connect to external resources on the robotic platform to perform various tasks related to robot perception and motor control. Given the blackboard architecture of the toolkit, the framework could however be extended to support fully distributed systems in the future.

7.1.2 System modules

System modules can operate in either synchronous and asynchronous mode:

- Synchronous modules continuously monitor the dialogue state for relevant changes. Their activation is thus in synchrony with the update events generated by the dialogue state.
- Asynchronous modules run independently of the dialogue state. They typically relate to visual or speech perception tasks. When new observations are made, they update the dialogue state with the corresponding information.

All modules have access to the complete dialogue state and can therefore exploit the full set of state variables (including generic contextual information) in their processing. We review below the modules shown in Figure 7.1 and describe their role and internal structure. It should be emphasised that the focus of the present thesis is on dialogue management. The other system modules are therefore deliberately limited to simple, "shallow" processing methods in order to concentrate

the implementation efforts on the dialogue manager. Many of these modules could however be extended to employ more sophisticated techniques, in particular in regard to speech recognition, natural language understanding and generation.

Speech recognition

Speech recognition is performed on the robot platform, using a commercial, off-the-shelf speech recognition engine (Vocon 3200 from Nuance). Four microphones placed on the robot head are used for the sound capture. The positioning of the microphones on the robot allows the user to interact with the robot in a natural manner, without needing to resort to head-mounted microphones. This increased naturalness comes however at the price of a degraded sound quality due to the larger (and varying) distance between the sound source and the microphones, combined with the presence of noisy mechanical motors at only a few centimetres from the sound capture.

The acoustic model employed in all experiments was provided along with the speech recognition engine, and was optimised for American English. The language model takes the form of context-free recognition grammars specified in Bachus-Naur Form (BNF). Distinct grammars were used to cover the domain of discourse of each experiment. The grammars were designed by hand, based on the Wizard-of-Oz transcripts collected in our empirical studies (cf. previous chapters). Grammars can be dynamically attached or removed from the engine at runtime, thereby allowing the system to adapt the recognition to the current dialogue context. This functionality is however not yet exploited in the current architecture (but see e.g. Lison (2010a) for a description of our prior research work on this issue). As the recognition engine only generates hypotheses with raw (unnormalised) confidence scores, a normalisation routine was implemented to transform them to a proper probability distribution $P(\tilde{u}_u)$.[1]

Natural language understanding

The goal of natural language understanding (NLU) is to map a collection of utterance hypotheses $\tilde{u}_u$ to a related set of dialogue act hypotheses $\tilde{a}_u$ expressing the semantic and pragmatic content of the user input. This understanding step is decomposed in our implementation in two tasks, dialogue act recognition and visual reference resolution.

The goal of dialogue act recognition is to construct the logical form representing the meaning of the utterance. It should be noted that a user utterance may contain more than one dialogue act, as for instance in "*yes and now pick the blue object*". A collection of domain-specific templates was designed by hand to convert surface forms into logical representations of dialogue acts. Although this approach does not allow for "deep" semantic extraction, it was shown to perform well in our dialogue domains. Future work may replace this template-based method with a data-driven semantic parser based on e.g. dependency parsing (Nivre et al., 2007).

Reference resolution is used to map linguistic expressions referring to objects in the visual context to their corresponding object identifier. This mapping is two steps. First, the properties stated in the linguistic expressions are matched against the set of possible references. In case the description remains ambiguous (i.e. more than one object matches the linguistic expression), the references can

[1] The conversion between confidence scores and probabilities is manually tuned in the current implementation. Future work may however rely on more principled estimation techniques such as the ones outlined in Williams (2008a).

be further ranked according to their visual saliency, defined here in terms of their physical distance to the robot.

Natural language understanding is practically implemented in openDial via probabilistic rules. As seen in Section 4.5, the formalism of probabilistic rules already includes special-purpose operators for string manipulation and can thus readily encode the shallow templates used for dialogue act recognition. Rule r_{15} below is an example of such rule. The rule lists three regular expression patterns associated with the dialogue act $\mathrm{MoveArm}(\mathrm{Left}, \mathrm{Down})$. If the value for the user utterance variable u_u matches at least one of the patterns, the dialogue act a_u is classified as $\mathrm{MoveArm}(\mathrm{Left}, \mathrm{Down})$:

$$
\begin{aligned}
r_{15}: \ \textbf{if} \ & (u_u \ \textit{matches} \text{ “ (*) left arm down”}) \\
& \vee (u_u \ \textit{matches} \text{ “ (*) lower (the}\,|\,\text{your) left arm”}) \\
& \vee (u_u \ \textit{matches} \text{ “ (*) down (the}\,|\,\text{your) left arm”}) \ \ \textbf{then} \\
& \Big\{ P\left(a_u = \mathrm{MoveArm}(\mathrm{Left}, \mathrm{Down})\right) = 1.0
\end{aligned}
$$

Dialogue management

Dialogue management follows the procedure outlined in Section 4.4 and will thus not be repeated here. After a dialogue state update, the dialogue manager triggers the corresponding rule-structured models, and selects the next action to perform (if any).

Natural language generation

If the selected system action is non-empty and corresponds to a verbal action, the natural language generation module is triggered. As for natural language understanding, the generation component of openDial is based on a manually designed collection of templates, but are here applied to convert a logical representation of the communicative goal into a surface form.

As for natural language understanding, the generation templates are also encoded with probabilistic rules – although this time the rules are utility rules, since generation is a decision-making task. As an example, rule r_{16} generates the system response u_m given the system act $a_m = \mathrm{Acknowledgement}$. The rule specifies in this case three alternatives with equal utility:

$$
\begin{aligned}
r_{16}: \ \textbf{if} \ & \left(a_m = \mathrm{Acknowledgement}\right) \ \ \textbf{then} \\
& \begin{cases}
U(u'_m = \text{“ok”}) = 1 \\
U(u'_m = \text{“great”}) = 1 \\
U(u'_m = \text{“thanks”}) = 1
\end{cases}
\end{aligned}
$$

The presence of multiple realisations allows for some variation in the system behaviour, since the system will automatically select one realisation at random (due to the equal utility assigned to the alternative realisations).

Speech synthesis

Speech synthesis is performed on the robot, using an off-the-shelf speech synthesis engine developed by Acapela[2]. The synthesis engine is based on unit selection. The output speech signal is then sent to two speakers placed on the robot head. To avoid spurious recognition results, the speech recognition is automatically disabled when the robot is speaking.

Robot perception

The robot can detect simple physical objects present in the visual scene. The object detection is done based on the vision libraries bundled with the robotic platform. Special markings are placed on top of the objects to facilitate the detection and the visual servoing.

Robot motion control

Various types of robot movements were employed in our experiments, including both generic body movements (rotating the arms and the head in various directions), spatial navigation (moving forward and backward, turning left and right) and object manipulation (grasping and releasing objects). All the movements were programmed by hand, using the motion control libraries on the robot.

7.1.3 Graphical user interface

The graphical user interface developed for the openDial toolkit allows the system designer to monitor and control in real-time the current state of the system. The interface is divided in two alternative views, shown as distinct tabs in the application window: the chat window and the dialogue state monitor.

Chat window

The chat window presents the interaction history as a chat window. The user inputs are shown as N-best lists together with their corresponding probabilities. Figure 7.2 provides a screenshot of the chat window.

In addition to monitoring the interactions, the chat window can also be used to test the dialogue system by typing new user and system inputs in the input field at the bottom of the window. The agent role can be switched in the drop-down field in the bottom right corner.

Dialogue state monitor

To allow the system designer to inspect the content of the dialogue state, a visualisation tool has also been integrated into openDial . The tool draws a graph with nodes corresponding to the state variables and directed edges corresponding to conditional dependencies.[3] An example of graph layout is shown in Figure 7.3. The graph is dynamically refreshed after each update of the dialogue state. The graph layout is automatically calculated to optimise the visualisation.

[2] http://www.acapela-group.com

[3] The graphs are rendered using JUNG, which is a Java-based open source toolkit for drawing various kinds of graph structures – cf. http://jung.sourceforge.net.

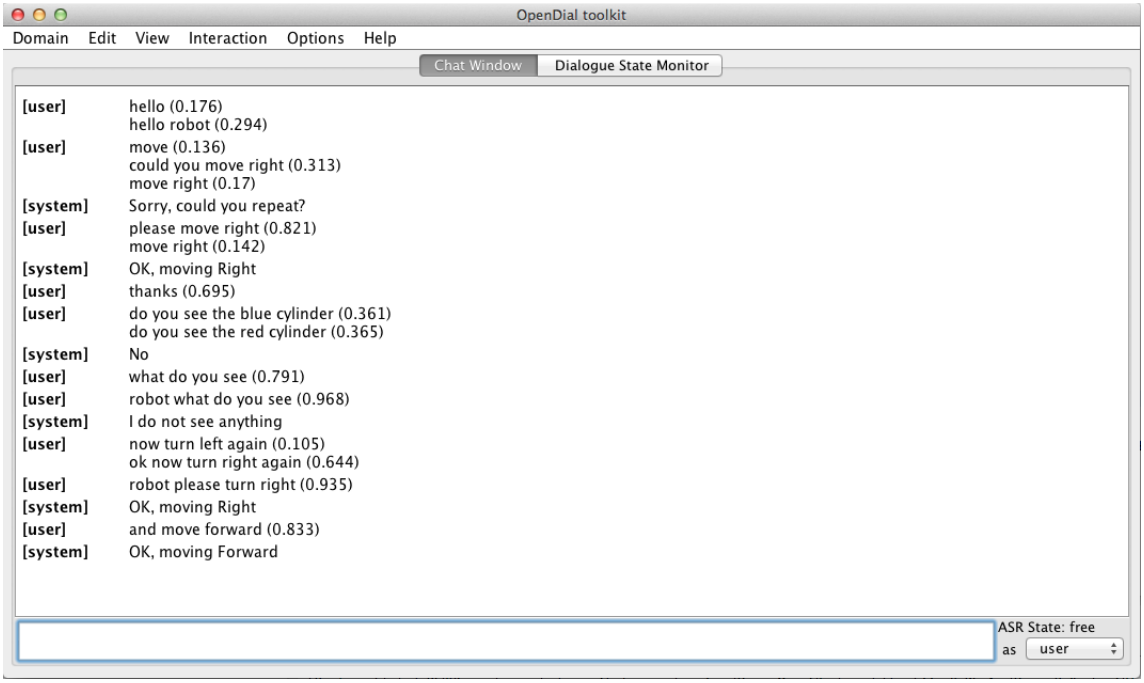


Figure 7.2: Graphical user interface showing the interaction history.

In addition to showing the current dialogue state, the dialogue state monitor can also record and store previous dialogue states. The dialogue state to visualise can be selected among the list on the left side of the window. This functionality is useful to e.g. compare dialogue states with one another and analyse how the dialogue state is evolving over time.

The graph can be manipulated in multiple ways in order to e.g. inspect the content of specific state variables, add or remove evidence, or request the calculation of marginal distributions on selected set of variables. The inference results are shown in the text area at the bottom of the window. In addition, the system designer can also directly view the shape of selected probability distributions using the distribution viewer tool illustrated in Figure 7.4. Discrete probability distributions are shown as histograms, while continuous probability distributions are graphically represented by their probability density functions.[4]

7.2 Specification of dialogue domains

7.2.1 Motivation

Many reasoning tasks can be structured in terms of probabilistic rules. As we have seen in the previous section, probabilistic rules have also been applied to natural language understanding and generation tasks in addition to dialogue management. The dialogue domain designed for the user experiments in Chapter 8 included for instance a total of 6 models: one dialogue act classification model, (triggered by the user utterance u_u), one action utility model (triggered by the user dialogue act a_u), three probability models to predict the effects of the system action on the context, the user intention and the next user action (all triggered by the system action a_m), and a generation model

[4]The graphical rendering of the probability distributions is done with the open source toolkit JFreeChart, cf. http://jfreechart.sourceforge.net.

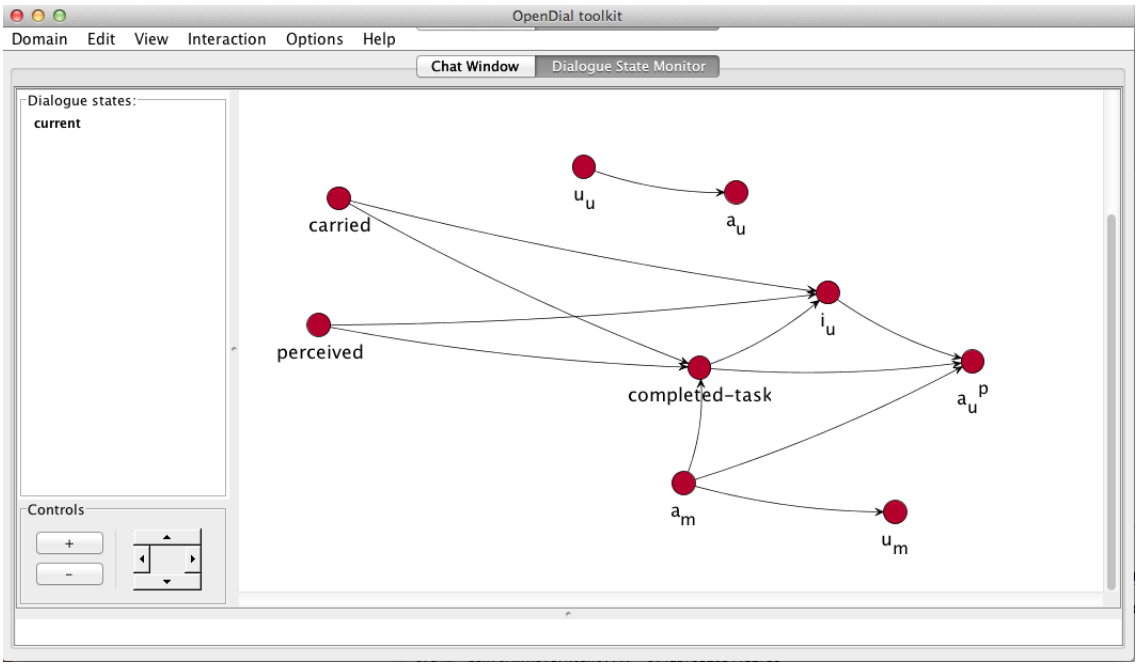


Figure 7.3: Visualisation of the current dialogue state.

(triggered by the system action a_m). The association of trigger variables to the rule-based models provides a simple and flexible a way to define the processing pipeline for the application. Variables can function as triggers for more than one model, allowing models to be instantiated in parallel.

As argued in Lison (2012a), the expressive power of probabilistic rules allow them to capture the structure of many dialogue processing tasks. Compared to traditional architectures in which the components are developed separately and rely on ad hoc representation formats, the use of a shared formalism to encode the domain models yields several advantages:

Transparency: The reliance on a common representation format provides a unified, transparent semantics for the dialogue state, since all state variables are described and related to one another through a principled framework grounded in probabilistic modelling. This makes it possible to derive a semantic interpretation for the dialogue state as a whole – in terms e.g. of a joint probability distribution over the state variables.

Domain portability: As all domain-specific knowledge is declaratively specified in the rules, the system architecture is essentially reduced to a generic platform for rule instantiation and probabilistic inference. This declarative design greatly enhances the system portability across domains, since adapting a system to a new domain only requires a rewrite or extension of the domain-specific rules, without having to reprogram a single component. This stands in sharp contrast with "black-box" types of architectures where much of the task- and domain-specific knowledge is encoded in procedural form within the component workflow.

Flexible workflow: Probability rules can design very flexible processing pipelines where state variables are allowed to depend or influence each other in any order and direction. Models can be easily inserted or extended without requiring any change to the underlying platform. Fur-

Figure 7.4: Distribution viewer showing both a discrete probability distribution for $P(a_u^p)$ and a continuous probability distribution $P(\theta_{15})$.

thermore, several models can be triggered concurrently on the same input/output variables.[5] This allows the system to take advantage of multiple, complementary modelling strategies while ensuring that the dialogue state remains consistent.

Joint optimisation: Finally, the use of a unified modelling formalism allows domain models to be optimised jointly instead of being tuned in isolation from one another. Joint optimisation has recently gained much attention in the dialogue system community to overcome the fragmentation of current system architectures and attempt to directly optimise the end-to-end conversational behaviour of the system (Lemon, 2011).

It should be noted that the architecture does not in any way preclude the integration of other types of processing modules in addition to rule-structured models, as long as these modules can read and update the dialogue state when relevant changes are detected.

7.2.2 Encoding format

Probabilistic rules are encoded in an XML format with a specifically designed syntax. Listing 7.1 illustrates an example of probability rule encoded in XML. Each rule is divided in cases, each

[5] Output distributions can indeed handle effect specifications arising from multiple, sometimes conflicting sources, as we have seen in Section 4.3.1.

containing a (possibly empty) condition and set of (possibly empty) effects. Conditions can include several sub-conditions combined as a conjunction or disjunction, the default being a conjunction. Logical variables are wrapped in curly brackets { } to distinguish them from normal values. Effects are associated with probabilities which can either be fixed or correspond to parameters to learn such as the Dirichlet distributions $\boldsymbol{\theta}_2$ and $\boldsymbol{\theta}_3$ in the example.

Utility rules are defined in the same manner. An example of utility rule specified in XML is given in Listing 7.2 for the rule r_1.

As explained in Section 4.4, a dialogue domain is defined as a pair $\langle \mathcal{B}_0, \mathcal{M} \rangle$, where $\mathcal{B}_0$ is the initial dialogue state and $\mathcal{M}$ the set of rule-based models attached to it. The specification of a complete dialogue domain thus takes the following form:

```
<domain>
   <initialstate>
         <!-- initial state variable values -->
   </initialstate>

   <model trigger="trigger variables for model 1">
         <!-- rules for model 1 -->
   </model>

   ...

   <model trigger="trigger variables for model n">
         <!-- rules for model n -->
   </model>

</domain>
```

Probability distributions defined in the initial state or the prior parameter distributions can be encoded either as categorical, Dirichlet, Gaussian or uniform distributions. As an illustration, the prior distribution for the parameter variable $\boldsymbol{\theta}_2 \sim \text{Dirichlet}(1, 2)$ is specified as:

```
<variable id="theta_2">
   <distrib type="dirichlet">
      <alpha>1</alpha>
      <alpha>2</alpha>
   </distrib>
</variable>
```

7.3 Core algorithms

7.3.1 Inference

switching algorithm

```
<rule>
   <quantifier id="O"/>
   <case>
      <condition>
         <if var="completed-task" value="true" />
         <if var="carried" value="{O}" relation="contains" />
      </condition>
      <effect prob="theta_2[0]">
         <set var="i_u" value="Release({O})" />
      </effect>
      <effect prob="theta_2[1]" />
   </case>
   <case>
      <condition>
         <if var="completed-task" value="true" />
      </condition>
      <effect prob="theta_3[0]">
         <set var="i_u" value="Release({O})" />
      </effect>
      <effect prob="theta_3[1]" />
   </case>
</rule>
```

Listing 7.1: Example of probability rule in XML format

```
<rule>
   <quantifier id="X"/>
   <case>
      <condition>
         <if var="silence" value="3" relation=">"/>
         <if var="a_m" value="Demonstrate({X})"/>
      </condition>
      <effect utility="theta_{confirmation3}">
         <set var="a_m" value="AskConfirmation"/>
      </effect>
   </case>
   <case>
      <condition>
         <if var="silence" value="2" relation=">"/>
         <if var="a_m" value="Demonstrate({X})"/>
      </condition>
      <effect utility="theta_{confirmation2}">
         <set var="a_m" value="AskConfirmation"/>
      </effect>
   </case>
   <case>
      <condition>
         <if var="silence" value="1" relation=">"/>
         <if var="a_m" value="Demonstrate({X})"/>
      </condition>
      <effect utility="theta_{confirmation1}">
         <set var="a_m" value="AskConfirmation"/>
      </effect>
   </case>
</rule>
```

Listing 7.2: Example of utility rule in XML format

7.3.2 Sampling techniques

7.3.3 Forward planning

switching algorithm, sampling methods for various distributions, kernel distributions anytime behaviour detail planning, state pruning

remarks on incrementality

7.4 Comparison with other architectures

Similarity to Olympus, Jaspis, trindikit, improTK, Ariadne dialogue architectures?

7.5 Conclusion

Chapter 8

User evaluation

ANOVA, since we would have two baselines and our approach? (see e.g. Passonneau's article)

Chapter 9

Concluding remarks

9.1 Summary of contributions

be able to capture richer conversational context, and better account for the cooperative nature of dialogue (e.g. Jokinen's argument against classical utility maximisation approaches).

9.2 Future work

Formally characterise the expressivity of the rules and extend them to handle Ginzburg style update rules?

more complex domains, with more variables and more complex dynamics

Try to learn a policy in a fully online fashion with real users, without simulator

do online reinforcement learning with real users and combine imitation+reinforcement learning. cognitive basis for this (toddlers). in line with Henderson's approach

joint optimisation of models

Appendix A

Relevant probability distributions

Uniform distribution

Multinomial & categorical distribution

Normal distribution

Dirichlet distribution

Kernel distribution

Should we include the last one?

Appendix B

Domain specifications

B.1 Experiments in Section 5.3

The dialogue management part of the domain is composed of a total of fifteen utility rules. The input variables of these rules are the user dialogue act a_u, the last system act a_m, the last physical movement $lastMove$, the recorded list of movements $sequence$, and a variable $silence$ expressing the amount of time elapsed since the last (user or system) action.

The variable a_u is encoded in this experiment as a list of one or more dialogue act. This encoding facilitates the processing of utterances containing more than one dialogue act.

r_1 : $\forall x$:

if $(silence > 3 \wedge a_m = Demonstrate(x))$ **then**

$\{U(a_m = AskConfirmation) = \theta_{\text{confirmation3}}$

else if $(silence > 2 \wedge a_m = Demonstrate(x))$ **then**

$\{U(a_m = AskConfirmation) = \theta_{\text{confirmation2}}$

else if $(silence > 1 \wedge a_m = Demonstrate(x))$ **then**

$\{U(a_m = AskConfirmation) = \theta_{\text{confirmation1}}$

r_2 : **if** $(Confirm \in a_u \wedge lastMove \neq None)$ **then**

$\{U(a_m = Register) = \theta_{\text{registerExplicit}}$

else

$\{U(a_m = Register) = \theta_{\text{registerExplicitNeg}}$

r_3 : **if** $(Disconfirm \in a_u \wedge None \neq lastMove)$ **then**

$\{U(a_m = Undo) = \theta_{\text{undo}}$

else

$\{U(a_m = Undo) = \theta_{\text{undoNeg}}$

r_4 : **if** $(a_u \neq None \wedge lastMove \notin a_u \wedge Confirm \notin a_u$

$\wedge Disconfirm \notin a_u \wedge lastMove \notin None$) **then**

$\Big\{ U(a_m = Register) = \theta_{\text{registerImplicit}}$

else

$\Big\{ U(a_m = Register) = \theta_{\text{registerImplicitNeg}}$

r_5 : $\forall x, y, z$:

if $(a_u = x \wedge lastMove \notin a_u \wedge (MoveArm(y, z) \in a_u$

$\vee MoveHead(y) \in a_u \vee Kneel \in a_u \vee MoveFoot(x, y) \in a_u$

$\vee StandUp \in a_u \vee SitDown \in a_u \vee Turn(y) \in a_u))$ **then**

$\Big\{ U(a_m = Demonstrate(x)) = \theta_{\text{Demonstrate}}$

else

$\Big\{ U(a_m = Demonstrate(x)) = \theta_{\text{DemonstrateNeg}}$

r_6 : **if** $(a_m \neq AskRepeat)$ **then**

$\Big\{ U(a_m = AskRepeat) = \theta_{\text{repeatFirst}}$

else

$\Big\{ U(a_m = AskRepeat) = \theta_{\text{repeatSecond}}$

r_7 : **if** $(Confirm \in a_u \wedge |a_u| = 1)$ **then**

$\Big\{ U(a_m = Ack) = \theta_{\text{ack}}$

else

$\Big\{ U(a_m = Ack) = \theta_{\text{ackNeg}}$

r_8 : **if** $(Disconfirm \in a_u \wedge |a_u| = 1)$ **then**

$\Big\{ U(a_m = AskIntention) = \theta_{\text{askIntention}}$

else

$\Big\{ U(a_m = AskIntention) = \theta_{\text{askIntentionNeg}}$

r_9 : **if** $(RepeatAll \in a_u \wedge sequence \neq None)$ **then**

$\Big\{ U(a_m = DemonstrateAll) = \theta_{\text{demonstrateAll}}$

r_{10} : **if** $(ForgetAll \in a_u \wedge sequence \neq None)$ **then**

$\Big\{ U(a_m = ForgetAll) = \theta_{\text{forgetAll}}$

r_{11} : **if** $(Compliment \in a_u \wedge a_m = DemonstrateAll)$ **then**

$\Big\{ U(a_m = SayThankYou) = \theta_{\text{sayThankYou}}$

r_{12} : **if** $(FollowMe \in a_u)$ **then**

$$\Big\{ U(a_m = FollowMe) = \theta_{\text{followMe}}$$

r_{13} : **if** $(Stop \in a_u)$ **then**

$$\Big\{ U(a_m = Stop) = \theta_{\text{stop}}$$

r_{14} : **if** $(SayHello \in a_u)$ **then**

$$\Big\{ U(a_m = SayHello) = \theta_{\text{sayHello}}$$

r_{15} : **if** $(SayGoodbye \in a_u)$ **then**

$$\Big\{ U(a_m = SayGoodbye) = \theta_{\text{sayGoodbye}}$$

B.2 Experiments in Section 6.3

First experiment

The six rules below specify the transition model used in the first experiment of Section 6.3. The first rule expresses the probability of the current task being completed after the last system action. If the task is completed, the user intention is reinitialised with a prior distribution given by rules r_3, r_4 and r_5. The user action model based on this user intention is then represented by rule r_6.

r_1: $\forall X$:

if $(a_m = Do(X) \wedge i_u = X)$ **then**

$$\Big\{ P(completed\text{-}task = true) = 1.0$$

else if $(a_m = Excuse(\cdot))$ **then**

$$\Big\{ P(completed\text{-}task = true) = 1.0$$

else if $(a_m = ConfirmDetection \wedge i_u = DoYouSee(\cdot))$ **then**

$$\Big\{ P(completed\text{-}task = true) = 1.0$$

else if $(a_m = DisconfirmDetection \wedge i_u = DoYouSee(\cdot))$ **then**

$$\Big\{ P(completed\text{-}task = true) = 1.0$$

else if $(a_m = Describe(\cdot) \wedge i_u = WhatDoYouSee)$ **then**

$$\Big\{ P(completed\text{-}task = true) = 1.0$$

else if $(a_m = Do(\cdot) \vee a_m = Excuse(\cdot) \vee a_m = Describe(\cdot)$

$\vee a_m = DisconfirmDetection \vee a_m = ConfirmDetection)$ **then**

$$\begin{cases} P(completed\text{-}task = true) = \theta_{1[0]} \\ P(completed\text{-}task = false) = \theta_{1[1]} \end{cases}$$

else if $(i_u = None)$ **then**

$$\left\{ P(\textit{completed-task} = \textit{true}) = 1.0 \right.$$

else

$$\left\{ P(\textit{completed-task} = \textit{false}) = 1.0 \right.$$

r_2: **if** $(\textit{completed-task} = \textit{false})$ **then**

$$\left\{ P(i_u = i_u) = 1 \right.$$

r_3: **if** $(\textit{completed-task} = \textit{true})$ **then**

$$\begin{cases} P(i_u = \textit{Move}(\textit{Left})) = \theta_{1[0]} \\ P(i_u = \textit{Move}(\textit{Forward})) = \theta_{2[1]} \\ P(i_u = \textit{Move}(\textit{Backward})) = \theta_{2[2]} \\ P(i_u = \textit{Move}(\textit{Right})) = \theta_{2[3]} \\ P(i_u = \textit{DoYouSee}(\textit{object}_1)) = \theta_{2[4]} \\ P(i_u = \textit{DoYouSee}(\textit{object}_2)) = \theta_{2[5]} \\ P(i_u = \textit{WhatDoYouSee}) = \theta_{2[6]} \\ P(\cdot) = \theta_{2[7]} \end{cases}$$

r_4: $\forall\, O:$

if $(\textit{completed-task} = \textit{true} \wedge O \in \textit{carried})$ **then**

$$\begin{cases} P(i_u = \textit{Release}(O)) = \theta_{3[0]} \\ P(\cdot) = \theta_{3[1]} \end{cases}$$

else if $(\textit{completed-task} = \textit{true})$ **then**

$$\begin{cases} P(i_u = \textit{Release}(O)) = \theta_{4[0]} \\ P(\cdot) = \theta_{4[1]} \end{cases}$$

r_5: $\forall\, O:$

if $(\textit{completed-task} = \textit{true} \wedge \textit{carried} = [])$ **then**

$$\begin{cases} P(i_u = \textit{PickUp}(O)) = \theta_{5[0]} \\ P(\cdot) = \theta_{5[1]} \end{cases}$$

else if $(\textit{completed-task} = \textit{true})$ **then**

$$\begin{cases} P(i_u = \textit{PickUp}(O)) = \theta_{6[0]} \\ P(\cdot) = \theta_{6[1]} \end{cases}$$

r_6: $\forall\, X:$

if $(a_m = \textit{Ground}(X) \wedge i_u = X)$ **then**

$$\begin{cases} P(a_u^p = \textit{Confirm}) = \theta_{7[0]} \\ P(a_u^p = \textit{Nothing}) = \theta_{7[1]} \\ P(\cdot) = \theta_{7[2]} \end{cases}$$

else if $(a_m = \textit{Ground}(X) \wedge i_u \neq X \wedge i_u \neq \textit{None})$ **then**

$$\begin{cases} P(a_u^p = Disconfirm) = \theta_{8[0]} \\ P(a_u^p = Ask(i_u)) = \theta_{8[1]} \\ P(a_u^p = Nothing) = \theta_{8[2]} \\ P(\cdot) = \theta_{8[3]} \end{cases}$$

else if $(a_m = Confirm(X) \wedge i_u = X)$ **then**

$$\begin{cases} P(a_u^p = Confirm) = \theta_{9[0]} \\ P(a_u^p = Ask(i_u)) = \theta_{9[1]} \\ P(a_u^p = Nothing) = \theta_{9[2]} \\ P(\cdot) = \theta_{9[3]} \end{cases}$$

else if $(a_m = Confirm(X) \wedge i_u \neq X \wedge i_u \neq None)$ **then**

$$\begin{cases} P(a_u^p = Disconfirm) = \theta_{10[0]} \\ P(a_u^p = Nothing) = \theta_{10[1]} \\ P(\cdot) = \theta_{10[2]} \end{cases}$$

else if $(a_m = Do(X) \wedge i_u = X)$ **then**

$$\begin{cases} P(a_u^p = RepeatLast) = \theta_{11[0]} \\ P(a_u^p = Ask(i_u)) = \theta_{11[1]} \\ P(a_u^p = Nothing) = \theta_{11[2]} \\ P(\cdot) = \theta_{11[3]} \end{cases}$$

else if $(a_m = AskRepeat \vee completed\text{-}task = true)$ **then**

$$\begin{cases} P(a_u^p = Ask(i_u)) = \theta_{12[0]} \\ P(a_u^p = Nothing) = \theta_{12[1]} \\ P(\cdot) = \theta_{12[2]} \end{cases}$$

else if $(i_u = None)$ **then**

$$\{P(a_u^p = Nothing) = 1.0$$

else

$$\begin{cases} P(a_u^p = Ask(i_u)) = \theta_{13[0]} \\ P(a_u^p = Nothing) = \theta_{13[1]} \\ P(\cdot) = \theta_{13[2]} \end{cases}$$

Second experiment

Two rule-structured model are compared to one another in the second experiment: a transition model for the model-based approach, and an action-value model for the model-free approach. The transition model is identical to the one shown above and will not be repeated here. The action-value model is specified with a set of 11 rules. The input variables of these rules are the last user dialogue act a_u, the user dialogue act preceding it $a_{u\text{-}prev}$, the last system act a_m, and the two contextual variables *perceived* and *carried*.

r_1: $\forall X$:

if $(a_u = Move(\cdot))$ **then**

$$\begin{cases} U(a_m = Do(a_u)) = \theta_{\text{movements1}} \\ U(a_m = Ground(a_u)) = \theta_{\text{movements2}} \end{cases}$$

else if $(a_u = Confirm \wedge a_{u\text{-}prev} = Move(\cdot))$ **then**

$$\begin{cases} U(a_m = Do(a_{u\text{-}prev})) = \theta_{\text{movements1}} \\ U(a_m = Ground(a_{u\text{-}prev})) = \theta_{\text{movements2}} \end{cases}$$

else if $(a_m = Ground(Move(X)) \wedge a_u \neq Disconfirm)$ **then**

$$\{U(a_m = Do(Move(X))) = \theta_{\text{movements1}}$$

r_2: $\forall X$:

if $(a_u = PickUp(X) \wedge X \in perceived \wedge carried = [])$ **then**

$$\begin{cases} U(a_m = Do(a_u)) = \theta_{\text{pickup1}} \\ U(a_m = Ground(a_u)) = \theta_{\text{pickup2}} \end{cases}$$

else if $(a_u = PickUp(X) \wedge X \in perceived \wedge carried \neq [])$ **then**

$$\{U(a_m = Excuse(AlreadyCarryObject)) = \theta_{\text{pickup3}}$$

else if $(a_u = PickUp(X) \wedge X \notin perceived)$ **then**

$$\{U(a_m = Excuse(DoNotSeeObject)) = \theta_{\text{pickup4}}$$

else if $(a_u = Confirm \wedge a_{u\text{-}prev} = PickUp(X) \wedge X \in perceived \wedge carried = [])$ **then**

$$\begin{cases} U(a_m = Do(a_{u\text{-}prev})) = \theta_{\text{pickup1}} \\ U(a_m = Ground(a_{u\text{-}prev})) = \theta_{\text{pickup2}} \end{cases}$$

else if $(a_u = Confirm \wedge a_{u\text{-}prev} = PickUp(X) \wedge X \in perceived \wedge carried \neq [])$ **then**

$$\{U(a_m = Excuse(AlreadyCarryObject)) = \theta_{\text{pickup3}}$$

else if $(a_u = Confirm \wedge a_{u\text{-}prev} = PickUp(X) \wedge X \notin perceived)$ **then**

$$\{U(a_m = Excuse(DoNotSeeObject)) = \theta_{\text{pickup4}}$$

else if $(a_m = Ground(PickUpX) \wedge a_u \neq Disconfirm \wedge X \in perceived \wedge carried = [])$ **then**

$$\{U(a_m = Do(PickUp(X))) = \theta_{\text{pickup1}}$$

else if $(a_m = Ground(PickUp(X)) \wedge a_u \neq Disconfirm \wedge X \in perceived \wedge carried \neq [])$ **then**

$$\{U(a_m = Excuse(AlreadyCarryObject)) = \theta_{\text{pickup3}}$$

else if $(a_m = Ground(PickUp(X)) \wedge a_u \neq Disconfirm \wedge X \notin perceived)$ **then**

$$\{U(a_m = Excuse(DoNotSeeObject)) = \theta_{\text{pickup4}}$$

r_3: $\forall X$:

if $(a_u = Release(X) \wedge X \in carried)$ **then**

$$\begin{cases} U(a_m = Do(a_u)) = \theta_{\text{release1}} \\ U(a_m = Ground(a_u)) = \theta_{\text{release2}} \end{cases}$$

else if $(a_u = Release(X) \wedge X \notin carried)$ **then**

$$\{U(a_m = Excuse(DoNotCarryObject)) = \theta_{\text{release3}}$$

else if $(a_u = Confirm \wedge a_{u\text{-}prev} = Release(X) \wedge X \in carried)$ **then**

$$\begin{cases} U(a_m = Do(a_{u\text{-}prev})) = \theta_{\text{release1}} \\ U(a_m = Ground(a_{u\text{-}prev})) = \theta_{\text{release2}} \end{cases}$$

else if $(a_u = Confirm \wedge a_{u\text{-}prev} = Release(X) \wedge X \notin carried)$ **then**

$$\begin{cases} U(a_m = Excuse(DoNotCarryObject)) = \theta_{\text{release3}} \end{cases}$$

else if $(a_m = Ground(Release(X)) \wedge a_u \neq Disconfirm \wedge X \in carried)$ **then**

$$\begin{cases} U(a_m = Do(a_u)) = \theta_{\text{release1}} \\ U(a_m = Ground(a_u)) = \theta_{\text{release2}} \end{cases}$$

else if $(a_m = Ground(Release(X)) \wedge a_u \neq Disconfirm \wedge X \notin carried)$ **then**

$$\begin{cases} U(a_m = Excuse(DoNotCarryObject)) = \theta_{\text{release3}} \end{cases}$$

r_4: **if** $(a_u \neq None)$ **then**

$$\begin{cases} U(a_m = None) = \theta_{\text{none}} \end{cases}$$

else

$$\begin{cases} U(a_m = None) = \theta_{\text{none2}} \end{cases}$$

r_5: **if** $(a_u \neq Confirm \wedge a_u \neq RepeatLast \wedge a_m \neq Ground(\cdot))$ **then**

$$\begin{cases} U(a_m = Do(\cdot) \wedge a_m \neq Do(a_u)) = \theta_{\text{wrong1}} \\ U(a_m = Ground(\cdot) \wedge a_m \neq Ground(a_u)) = \theta_{\text{wrong2}} \\ U(a_m = Excuse(\cdot)) = \theta_{\text{wrong3}} \end{cases}$$

else if $(a_u = Confirm)$ **then**

$$\begin{cases} U(a_m = Do(\cdot) \wedge a_m \neq Do(a_{u\text{-}prev})) = \theta_{\text{wrong1}} \\ U(a_m = Ground(\cdot) \wedge a_m \neq Ground(a_{u\text{-}prev})) = \theta_{\text{wrong2}} \\ U(a_m = Excuse(\cdot)) = \theta_{\text{wrong3}} \end{cases}$$

r_6: **if** $(a_m \neq AskRepeat)$ **then**

$$\begin{cases} U(a_m = AskRepeat) = \theta_{\text{repeat}} \end{cases}$$

r_7: **if** $(a_u = Disconfirm)$ **then**

$$\begin{cases} U(a_m = AskRepeat) = \theta_{\text{repeat2}} \end{cases}$$

r_8: **if** $(true)$ **then**

$$\begin{cases} U(a_m = Confirm(a_u)) = \theta_{\text{confirm1}} \\ U(a_m = Confirm(\cdot) \wedge a_m \neq Confirm(a_u)) = \theta_{\text{confirm2}} \end{cases}$$

r_9: $\forall X$:

if $(a_u = DoYouSee(X) \wedge X \in perceived)$ **then**

$$\begin{cases} U(a_m = Confirm) = \theta_{\text{doyousee1}} \\ U(a_m = Disconfirm) = \theta_{\text{doyousee2}} \end{cases}$$

else if $(a_u = DoYouSee(X) \wedge X \notin perceived)$ **then**

$$\begin{cases} U(a_m = Disconfirm) = \theta_{\text{doyousee3}} \\ U(a_m = Confirm) = \theta_{\text{doyousee4}} \end{cases}$$

else if $(a_u = Confirm \wedge a_{u\text{-}prev} = DoYouSee(X) \wedge X \in perceived)$ **then**

$$\begin{cases} U(a_m = Confirm) = \theta_{\text{doyousee1}} \\ U(a_m = Disconfirm) = \theta_{\text{doyousee2}} \end{cases}$$

else if $(a_u = Confirm \wedge a_{u\text{-}prev} = DoYouSee(X) \wedge X \notin perceived)$ **then**

$$\begin{cases} U(a_m = Disconfirm) = \theta_{\text{doyousee3}} \\ U(a_m = Confirm) = \theta_{\text{doyousee4}} \end{cases}$$

else

$$\begin{cases} U(a_m = Confirm) = \theta_{\text{doyousee5}} \\ U(a_m = Disconfirm) = \theta_{\text{doyousee6}} \end{cases}$$

r_{10}: **if** $(a_u = WhatDoYouSee)$ **then**

$$\{U(a_m = Describe(perceived)) = \theta_{\text{whatdoyousee1}}$$

else if $(a_u = Confirm \wedge a_{u\text{-}prev} = WhatDoYouSee)$ **then**

$$\{U(a_m = Describe(perceived)) = \theta_{\text{whatdoyousee1}}$$

else

$$\{U(a_m = Describe(\cdot)) = \theta_{\text{whatdoyousee2}}$$

r_{11}: **if** $(a_u = RepeatLast)$ **then**

$$\{U(a_m = a_m) = \theta_{\text{repeatLast}}$$

B.3 Experiments in Chapter 8

put here a summary of the probabilistic rules applied in the last experiment (user evaluation)

Bibliography

J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. An architecture for a generic dialogue shell. *Natural Language Engineering*, 6:213–228, 2000.

J. Allen, G. Ferguson, and A. Stent. An architecture for more realistic conversational systems. In *Proceedings of the 6th international conference on Intelligent user interfaces (IUI 2001)*, pages 1–8, New York, NY, USA, 2001. ACM.

J. F. Allen and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15:143–178, 1980.

J. Allwood, J. Nivre, and E. Ahlsén. On the semantics and pragmatics of linguistic feedback. *Journal of Semantics*, 9:1–26, 1992.

J. S Allwood. *Linguistic communication as action and cooperation : a study in pragmatics*. PhD thesis, Dept. of Linguistics, University of Göteborg, 1976.

N. Asher and A. Lascarides. *Logics of Conversation*. Cambridge University Press, Cambridge, 2005.

Amin Atrash and Joelle Pineau. A bayesian reinforcement learning approach for customizing human-robot interfaces. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, pages 355–360. ACM, 2009.

J. L. Austin. *How to do things with words*. Harvard University Press, Cambridge, Mass., 1962.

J. B Bavelas, A. Black, C. R. Lemery, and J. Mullett. "I show how you feel": Motor mimicry as a communicative act. *Journal of Personality and Social Psychology*, 50(2):322–329, 1986.

J. Baxter and P. L. Bartlett. Infinite-horizon gradient-based policy search. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

R.E. Bellman. *Dynamic programming*. Princeton University Press, Princeton, NY, 1957.

Y. Bengio. Learning deep architectures for ai. *Foundational Trends in Machine Learning*, 2(1): 1–127, January 2009.

C. L. Bennett and A. I. Rudnicky. The Carnegie Mellon COMMUNICATOR corpus. In J. H. L. Hansen and B. L. Pellom, editors, *INTERSPEECH*. ISCA, 2002.

L. Benotti. *Implicature as an Interactive Process*. PhD thesis, Université Henri Poincaré, Nancy, 2010.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition, 1996.

T. W. Bickmore and T. Giorgino. Health dialog systems for patients and consumers. *Journal of Biomedical Informatics*, pages 556–571, 2006.

J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2-3):213–244, 1997.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

D. Bohus and A. I. Rudnicky. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23(3):332–361, July 2009a.

D. Bohus and A. I. Rudnicky. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23:332–361, 2009b.

J. Bos, E. Klein, O. Lemon, and T. Oka. DIPPER: Description and formalisation of an information-state update dialogue system architecture. In *4th SIGdial Workshop on Discourse and Dialogue - remember to check the ACL anthology to find the correct conference names*, pages 115–124, 2003.

A. Boularias, H. R. Chinaei, and B. Chaib-draa. Learning the reward model of dialogue pomdps. In *Proceedings of the NIPS Workshop on Machine Learning for Assistive Technology (MLAT 2010)*, 2010.

C. Boutilier, T. Dean, and S. Hanks. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

H. P. Branigan, M. J. Pickering, and A. A. Cleland. Syntactic co-ordination in dialogue. *Cognition*, 75(2):B13–B25, 2000.

S. E. Brennan and H. H. Clark. Conceptual pacts and lexical choice in conversation. *Journal of Experimental Psychology-Learning Memory and Cognition*, 22(6):1482–1493, 1996.

M. Buckley and C. Benzmüller. An agent-based architecture for dialogue systems. In *Proceedings of the 6th international Andrei Ershov memorial conference on Perspectives of systems informatics*, PSI'06, pages 135–147, Berlin, Heidelberg, 2007. Springer-Verlag.

T. H. Bui, J. Zwiers, M. Poel, and A. Nijholt. Affective dialogue management using factored POMDPs. In R. Babuska and F.C.A. Groen, editors, *Interactive Collaborative Information Systems*, volume 281 of *Studies in Computational Intelligence*, pages 209–238. Springer Verlag, March 2010.

H. C. Bunt. Dynamic Interpretation and Dialogue Theory. In M. M. Taylor, F. Néel, and D. G. Bouwhuis, editors, *The Structure of Multimodal Dialogue, Volume 2*. John Benjamins, 1996.

L. Burnard. User reference guide for the british national corpus. Technical report, Oxford University, 2000.

R. Cantrell, M. Scheutz, P. W. Schermerhorn, and X. Wu. Robust spoken instruction understanding for HRI. In *HRI*, pages 275–282. ACM, 2010.

S. Castronovo, A. Mahr, M. Pentcheva, and C. A. Müller. Multimodal dialog in the car: combining speech and turn-and-push dial to control comfort functions. In Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura, editors, *Proceeding of Interspeech*, pages 510–513. ISCA, 2010.

L. Chen, A. Wang, and B. Di Eugenio. Improving pronominal and deictic co-reference resolution with multi-modal features. In *Proceedings of the SIGDIAL 2011 Conference*, pages 307–311. Association for Computational Linguistics, 2011.

S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393, 1999.

J. Cheng and M. J. Druzdzel. Ais-bn: An adaptive importance sampling algorithm for evidential reasoning in large bayesian networks. *Journal of Artificial Intelligence Research*, 13(1):155–188, 2000.

M. Chi, K. Van Lehn, D. J. Litman, and P. W. Jordan. An evaluation of pedagogical tutorial tactics for a natural language tutoring system: A reinforcement learning approach. *International Journal of Artificial Intelligence in Education*, 21(1-2):83–113, 2011.

H. R. Chinaei and B. Chaib-draa. An inverse reinforcement learning algorithm for partially observable domains with application on healthcare dialogue management. In *ICMLA (1)*, pages 144–149. IEEE, 2012.

H. R. Chinaei, B. Chaib-draa, and L. Lamontagne. Learning observation models for dialogue POMDPs. In L. Kosseim and D. Inkpen, editors, *Advances in Artificial Intelligence*, volume 7310 of *Lecture Notes in Computer Science*, pages 280–286. Springer Berlin Heidelberg, 2012.

J. Choi and K.-E. Kim. Inverse reinforcement learning in partially observable environments. *Journal of Maching Learning Research*, 12:691–730, July 2011.

H. H. Clark. *Using Language*. Cambridge: Cambridge University Press, 1996.

H. H. Clark and E. F. Schaefer. Contributing to discourse. *Cognitive Science*, 13(2):259–294, 1989.

P. R. Cohen and C. R. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.

G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

R. Cooper. Type theory and semantics in flux. In Tim Fernando Ruth Kempson and Nicholas Asher, editors, *Handbook of the Philosophy of Science, Volume 14: Philosophy of Linguistics*. Elsevier, 2012.

B. Coppola, A. Moschitti, and G. Riccardi. Shallow semantic parsing for spoken language understanding. In *Proceedings of Human Language Technologies: The 10th meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2010)*, pages 85–88. Association for Computational Linguistics, 2009.

J. G. Core and J. F. Allen. Coding dialogs with the DAMSL annotation scheme. In *Proceedings of the Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*, Cambridge, MA, November 1997.

P. A. Crook and O. Lemon. Representing uncertainty about complex user goals in statistical dialogue systems. In *Proceedings of the 11th SIGDIAL meeting on Discourse and Dialogue*, pages 209–212, 2010.

P. A. Crook and O. Lemon. Lossless value directed compression of complex user goal states for statistical spoken dialogue systems. In ISCA, editor, *INTERSPEECH*, pages 1029–1032, 2011.

H. Cuayáhuitl. Learning Dialogue Agents with Bayesian Relational State Representations. In *Proceedings of the IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems (IJCAI-KRPDS)*, Barcelona, Spain, 2011.

H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira. Evaluation of a hierarchical reinforcement learning spoken dialogue system. *Computer Speech & Language*, 24:395–429, 2010.

P. Dagum and M. Luby. Approximating probabilistic inference in bayesian belief networks is np-hard. *Artificial Intelligence*, 60(1):141 – 153, 1993.

N. Dahlbäck, A. Jönsson, and Lars Ahrenberg. Wizard of oz studies: why and how. In *Proceedings of the 1st International Conference on Intelligent User Interfaces*, pages 193–200. ACM, 1993.

L. Daubigney, M. Geist, S. Chandramohan, and O. Pietquin. A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimisation. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):891–902, December 2012a.

L. Daubigney, M. Geist, and O. Pietquin. Off-policy learning in large-scale POMDP-based dialogue systems. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4989 –4992, 2012b.

R. De Mori, F. Bechet, D. Hakkani-Tur, M. McTear, G. Riccardi, and G. Tur. Spoken language understanding. *Signal Processing Magazine, IEEE*, 25(3):50–58, 2008.

R. Dearden, N. Friedman, and S. Russell. Bayesian q-learning. In *Proceedings of the fifteenth national conference on Artificial intelligence (AAAI 1998)*, pages 761–768, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

R. Dearden, N. Friedman, and D. Andre. Model-based Bayesian Exploration. In K. B. Laskey and H. Prade, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI 1999)*, pages 150–159, 1999.

N. Dethlefs and H. Cuayáhuitl. Hierarchical reinforcement learning and hidden markov models for task-oriented natural language generation. In *ACL (Short Papers)*, pages 654–659. The Association for Computer Linguistics, 2011.

F. J. Díez and M. J. Druzdzel. Canonical probabilistic models for knowledge engineering. Technical Report CISIAD-06-01, UNED, Madrid, Spain, 2006.

F. Doshi and N. Roy. Spoken language interaction with model uncertainty: an adaptive human-robot interaction system. *Connection Science*, 20(4):299–318, 2008a.

Finale Doshi and N. Roy. Spoken language interaction with model uncertainty: an adaptive human-robot interaction system. *Connection Science*, 20(4):299–318, December 2008b.

M. Duff. *Optimal learning: computational procedures for bayes-adaptive markov decision processes*. PhD thesis, University of Massachusetts Amherst, 2002.

S. Duncan. Some signals and rules for taking speaking turns in conversations. *Journal of Personality and Social Psychology*, 23:283–292, 1972.

M. O. Dzikovska, A. Isard, P. Bell, J. D. Moore, N. B. Steinhauser, G. E. Campbell, L. S. Taylor, S. Caine, and C. Scott. Adaptive intelligent tutorial dialogue in the BEETLE II system. In *Proceedings of the 15th international conference on Artificial intelligence in education*, AIED'11, pages 621–621, Berlin, Heidelberg, 2011. Springer-Verlag.

M. Eckert and M. Strube. Dialogue acts, synchronizing units, and anaphora resolution. *Journal of Semantics*, 17(1):51–89, 2000.

P. Ehlen and M. Johnston. A multimodal dialogue interface for mobile local search. In J. Kim, J. Nichols, and P. A. Szekely, editors, *IUI Companion*, pages 63–64. ACM, 2013.

Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with gaussian processes. In *Proceedings of the 22nd international conference on Machine learning (ICML 2005)*, pages 201–208, New York, NY, USA, 2005. ACM.

R. Fernández. *Non-Sentential Utterances in Dialogue: Classification, Resolution and Use*. PhD thesis, Ph. D. thesis, King?s College, London. 935, 2006.

R. Fernández, J. Ginzburg, and S. Lappin. Classifying non-sentential utterances in dialogue: A machine learning approach. *Computational Linguistics*, 33(3):397–427, September 2007.

M. Frampton and O. Lemon. Recent research advances in reinforcement learning in spoken dialogue systems. *Knowledge Engineering Review*, 24(4):375–408, 2009.

M. Frampton, R. Fernández, P. Ehlen, M. Christoudias, T. Darrell, and S. Peters. Who is "you"?: combining linguistic and gaze features to resolve second-person references in dialogue. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 273–281. Association for Computational Linguistics, 2009.

R. Freedman. Plan-based dialogue management in a physics tutor. In *Proceedings of the sixth conference on Applied natural language processing*, ANLC '00, pages 52–59. Association for Computational Linguistics, 2000.

J. Fritsch, M. Kleinehagenbrock, A. Haasch, S. Wrede, and G. Sagerer. A flexible infrastructure for the development of a robot companion with extensible hri-capabilities. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, pages 3408–3414, 2005.

K. Funakoshi, M. Nakano, T. Tokunaga, and R. Iida. A unified probabilistic approach to referring expressions. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '12, pages 237–246. Association for Computational Linguistics, 2012.

R. M. Fung and K.-C. Chang. Weighing and integrating evidence for stochastic simulation in bayesian networks. In *Proceedings of the 5th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 209–220, 1989.

D. Gamerman and H. F. Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*, volume 68. Chapman & Hall/CRC, 2006.

L.T.F. Gamut. *Logic, language, and meaning: Introduction to logic. Volume 1*. Logic, language, and meaning. University of Chicago Press, 1991.

S. Garrod and M. J. Pickering. Joint action, interactive alignment, and dialog. *Topics in Cognitive Science*, 1(2):292–304, 2009.

S. Garrod and M.J. Pickering. Why is conversation so easy? *Trends in Cognitive Sciences*, 8:8–11, 2004.

M. Gašić, F. Jurčíček, B. Thomson, Kai Yu, and S. Young. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 312–317, 2011.

M. Geist and O. Pietquin. Kalman temporal differences. *Journal of Artificial Intelligence Research*, 39:483–532, 2010.

K. Georgila, J. Henderson, and O. Lemon. User simulation for spoken dialogue systems: learning and evaluation. In *Proceedings of the Ninth International Conference on Spoken Language Processing (INTERSPEECH-ICSLP 2006)*, 2006.

L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

M. Ghavamzadeh and Y. Engel. Bayesian policy gradient algorithms. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *NIPS*, pages 457–464. MIT Press, 2006. ISBN 0-262-19568-2.

H. Giles, N. Coupland, and J. Coupland. 1. accommodation theory: Communication, context, and consequence. *Contexts of accommodation: Developments in applied sociolinguistics*, page 1, 1991.

J. Ginzburg. *The Interactive Stance*. Oxford University Press, New York, 2012.

J. J. Godfrey, E. C. Holliman, and J. McDaniel. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of ICASSP*, volume 1, pages 517–520 vol.1, 1992.

M. A. Goodrich and A. C. Schultz. Human-robot interaction: a survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.

A. L. Gorin, G. Riccardi, and J. H. Wright. How may i help you? *Speech Communication*, 23(1-2): 113–127, 1997.

A. Gravano and J. Hirschberg. Turn-taking cues in task-oriented dialogue. *Computer Speech & Language*, 25(3):601 – 634, 2011.

P. J. Green. On use of the EM algorithm for penalized likelihood estimation. *Journal of the Royal Statistical Society*, 52(3):443–452, 1990.

H.P. Grice. *Studies in the Way of Words*. Harvard University Press, 1989.

B. J. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12:175–204, July 1986.

A. Gruenstein, C. Wang, and S. Seneff. Context-sensitive statistical language modeling. In *Proceedings of 9th European Conference on Speech Communication and Technology (Interspeech 2005)*, pages 17–20, 2005.

E. A. Hansen. Solving POMDPs by searching in policy space. In *UAI*, pages 211–219, 1998.

J. H. L. Hansen, X. Zhang, M. Akbacak, U.H. Yapanel, B. Pellom, W. Ward, and P. Angkititrakul. CU-move: Advanced in-vehicle speech systems for route navigation. In H. Abut, J. H . L. Hansen, and K. Takeda, editors, *DSP for In-Vehicle and Mobile Systems*, pages 19–45. Springer, 2005.

M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. Dean, and C. Boutilier. Hierarchical solution of markov decision processes using macro-actions. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 220–229, 1998.

N. Hawes, A. Sloman, J. L. Wyatt, M. Zillich, H. Jacobsson, G.-J. M. Kruijff, M. Brenner, G. Berginc, and D. Skocaj. Towards an integrated robot with multiple cognitive functions. In *AAAI*, pages 1548–1553. AAAI Press, 2007.

Y. He and S. Young. Semantic processing using the hidden vector state model. *Computer Speech & Language*, 19(1):85 – 106, 2005.

P. Heeman. Combining reinforcement learning with Information-State update rules. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 268–275, 2007.

J. Henderson, O. Lemon, and K. Georgila. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 34:487–511, 2008.

G. Herzog, A. Ndiaye, S. Merten, H. Kirchmann, T. Becker, and P. Poller. Large-scale software integration for spoken language and multimodal dialog systems. *Natural Language Engineering*, 10(3-4):283–305, September 2004.

J.-H. Hong, Y.-S. Song, and S.-B. Cho. Mixed-initiative human–robot interaction using hierarchical bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 37(6):1158–1164, November 2007.

L. Horn and G. Ward. *Handbook of pragmatics*, volume 26. Wiley-Blackwell, 2008.

E. Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '99, pages 159–166, New York, NY, USA, 1999. ACM.

A. J. Hunt and A. W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Proceedings of the 21st International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1996)*, volume 1, pages 373–376 vol. 1, 1996.

L. F. Hurtado, D. Griol, E. Sanchis, and E. Segarra. A stochastic approach to dialog management. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2005)*, pages 226–231. IEEE, 2005.

M. Jaeger. Complex probabilistic modeling with recursive relational bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32(1-4):179–220, 2001.

D. Jan, E. Chance, D. Rajpurohit, D. DeVault, A. Leuski, J. Morie, and D. Traum. Checkpoint exercise: Training with virtual actors in virtual worlds. In *Intelligent Virtual Agents*, volume 6895 of *Lecture Notes in Computer Science*, pages 453–454. Springer, 2011.

S. Janarthanam, O. Lemon, X. Liu, P. Bartie, W. Mackaness, T. Dalmas, and J. Goetze. Integrating location, visibility, and question-answering in a spoken dialogue system for pedestrian city exploration. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 134–136. Association for Computational Linguistics, 2012.

F. Jelinek. *Statistical methods for speech recognition*. MIT Press, Cambridge, MA, USA, 1997.

F. V. Jensen, K. G. Olesen, and S. K. Andersen. An algebra of bayesian belief universes for knowledge-based systems. *Networks*, 20(5):637–659, 1990.

M. Johnson and E. Charniak. A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, ACL '04. Association for Computational Linguistics, 2004.

K. Jokinen. *Constructive Dialogue Modelling: Speech Interaction and Rational Agents*. Wiley-Interscience, New York, NY, USA, 2009.

K. Jokinen and T. Hurtig. User expectations and real experience on a multimodal interactive system. In *INTERSPEECH*. ISCA, 2006.

A. Jönsson and N. Dahlbäck. Talking to a computer is not like talking to your best friend. In *Proceedings of the First Scandinavian Conference on Artificial Intelligence*, pages 53–68, 1988.

M. Jordan. *Learning in Graphical Models*. The MIT Press, 1998.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, November 1999.

D. Jurafsky, L. Shriberg, and D. Biasca. Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual, draft 13. Technical report, University of Colorado at Boulder Technical Report 97-02, 1997.

D. Jurafsky, E. Shriberg, B. Fox, and T. Curl. Lexical, prosodic, and syntactic cues for dialog acts. In *Proceedings of ACL/COLING-98 Workshop on Discourse Relations and Discourse Markers*, pages 114–120, 1998.

D. Kahneman, P. Slovic, and A. Tversky. *Judgement under uncertainty - Heuristics and biases*. Cambridge University Press, Cambridge, 1981.

M. Kearns. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *Machine Learning*, pages 1324–1331, 1999.

S. Keizer and R. op den Akker. Dialogue act recognition under uncertainty using Bayesian networks. *Natural Language Engineering*, 13:287–316, 11 2007.

J. Kelleher and J. Van Genabith. Visual salience and reference resolution in simulated 3-d environments. *Artificial Intelligence Review*, 21(3-4):253–267, June 2004.

C. Kennington and D. Schlangen. Markov logic networks for situated incremental natural language understanding. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2012)*, pages 314–323, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

K. Kersting and L. De Raedt. Logical markov decision programs and the convergence of logical td(lambda). In *ILP*, volume 3194, pages 180–197. Springer, 2004.

S. Kok and P. Domingos. Learning markov logic network structure via hypergraph lifting. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009)*, pages 505–512, New York, NY, USA, 2009. ACM.

A. Koller and M. Stone. Sentence generation as a planning problem. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 336–343, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

A. Koller, K. Garoufi, M. Staudte, and M. W. Crocker. Enhancing referential success by tracking hearer gaze. In *SIGDIAL Conference*, pages 30–39, 2012.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

K. Komatani, K. Tanaka, H. Kashima, and T. Kawahara. Domain-independent spoken dialogue platform using key-phrase spotting based on combined language model. In P. Dalsgaard, B. Lindberg, H. Benner, and Z.-Hua Tan, editors, *INTERSPEECH*, pages 1319–1322. ISCA, 2001.

G.-J. M. Kruijff, P. Lison, T. Benjamin, H. Jacobsson, H. Zender, I. Kruijff-Korbayovà, and N. Hawes. Situated dialogue processing for human-robot interaction. In H. Christensen, G.-J. M.. Kruijff, and J. L. Wyatt, editors, *Cognitive Systems*, pages 311–364. Springer Verlag, 2010.

S. Kullback and R. A. Leibler. On Information and Sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

H. Kurniawati, D. Hsu, and W.S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science and Systems*, 2008.

I. R. Lane, T. Kawahara, and S. Ueno. Example-based training of dialogue planning incorporating user and situation models. In *INTERSPEECH*. ISCA, 2004.

T. Lang and M. Toussaint. Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research*, 39:1–49, 2010.

S. Larsson. *Issue-based Dialogue Management*. PhD thesis, Gothenburg University, 2002.

S. Larsson and D. R. Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natuarl Language Engineering*, 6(3-4):323–340, September 2000a.

S. Larsson and D. R. Traum. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural Language Engineering*, 6:323–340, September 2000b.

S. Larsson and D. R. Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural language engineering*, 6(3 & 4):323–340, 2000c.

S. Larsson, R. Cooper, E. Engdahl, and P. Ljunglöf. Information states and dialogue move engines. *Electronic Transactions on Artificial Intelligence*, 3(D):53–71, 1999.

S. Lemaignan, R. Ros, E. A. Sisbot, R. Alami, and M. Beetz. Grounding the interaction: Anchoring situated discourse in everyday human-robot interaction. *International Journal of Social Robotics*, 4(2):181–199, 2012.

O. Lemon. Learning what to say and how to say it: Joint optimisation of spoken dialogue management and natural language generation. *Computer Speech & Language*, 25:210–221, 2011.

O. Lemon and O. Pietquin. Machine Learning for Spoken Dialogue Systems. In *Proceedings of the 10th European Conference on Speech Communication and Technologies (Interspeech'07)*, pages 2685–2688, 2007.

O. Lemon, K. Georgila, and J. Henderson. Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: the TALK TownInfo evaluation. In *Spoken Language Technology Workshop, 2006. IEEE*, pages 178–181, 2006.

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

S. C. Levinson. *Pragmatics*. Cambridge University Press, 1983.

P. Lison. A salience-driven approach to speech recognition for human-robot interaction. In *Interfaces: Explorations in Logic, Language and Computation*, pages 102–113. Springer Verlag, 2010a.

P. Lison. Towards relational POMDPs for adaptive dialogue management. In *Proceeding of the Student Research Workshop of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010b.

P. Lison. Multi-policy dialogue management. In *Proceedings of the 12th SIGDIAL Meeting on Discourse and Dialogue*, Portland, USA, 2011.

P. Lison. Declarative design of spoken dialogue systems with probabilistic rules. In *Proceedings of the 16th Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2012)*, pages 97–106, 2012a.

P. Lison. Towards online planning for dialogue management with rich domain knowledge. In *Natural Interaction with Robots, Knowbots and Smartphones - Putting Spoken Dialog Systems into practice (IWSDS 2012)*. Springer, November 2012b.

P. Lison. Towards dialogue management in relational domains. In *SLTC Workshop on Action, Perception and Language (APL)*, Lund, Sweden, October 2012c.

P. Lison. Probabilistic dialogue models with prior domain knowledge. In *Proceedings of the SIGDIAL 2012 Conference*, pages 179–188, 2012d.

P. Lison. Model-based bayesian reinforcement learning for dialogue management. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association (Interspeech 2013)*, 2013.

D. J. Litman and J. F. Allen. A plan recognition model for subdialogues in conversations. *Cognitive Science*, 11(2):163–200, 1987.

M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: scaling up. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in agents*, pages 495–503. Morgan Kaufmann, San Francisco, CA, USA, 1998.

R. López-Cózar and Z. Callejas. Multimodal dialogue for ambient intelligence and smart environments. In H. Nakashima, H. Aghajan, and J.-C. Augusto, editors, *Handbook of Ambient Intelligence and Smart Environments*, pages 559–579. Springer US, 2010.

D. J.C. MacKay. Introduction to monte carlo methods. In *Learning in graphical models*, pages 175–204. Springer, 1998.

C. Matheson, M. Poesio, and D. R. Traum. Modelling grounding and discourse obligations using update rules. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference (NAACL 2000)*, pages 1–8, 2000.

D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL - the planning domain definition language. Technical Report CVC TR98003 / DCS TR1165, Yale Center for Computational Vision and Control, 1998.

M. F. McTear. *Spoken dialogue technology - toward the conversational user interface*. Springer, 2004.

N. Mehta, R. Gupta, A. Raux, D. Ramachandran, and S. Krawczyk. Probabilistic ontology trees for belief tracking in dialog systems. In *Proceedings of the 11th SIGDIAL Meeting on Discourse and Dialogue*, pages 37–46, 2010.

J.-J. Ch. Meyer and W. Van Der Hoek. *Epistemic logic for AI and computer science*, volume 41. Cambridge University Press, 2004.

T.P. Minka. Estimating a Dirichlet distribution. *Annals of Physics*, 2000(8):1–13, 2003.

N. Mirnig, A. Weiss, G. Skantze, S. Al Moubayed, J. Gustafson, J. Beskow, B. Granström, and M. Tscheligi. Face-to-face with a robot: What do we actually talk about? *International Journal of Humanoid Robotics*, 10(1), 2013.

F. Morbini, E. Forbell, D. DeVault, K. Sagae, D. R. Traum, and A. A. Rizzo. A mixed-initiative conversational dialogue system for healthcare. In *SIGDIAL Conference*, pages 137–139. The Association for Computer Linguistics, 2012.

M. G. Morgan and M. Henrion. *Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*. Cambridge University Press, 1992.

K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence (UAI 1999)*, pages 467–475, San Francisco, CA, USA, 1999.

Y. I. Nakano, G. Reinstein, T. Stocky, and J. Cassell. Towards a model of face-to-face grounding. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 553–561. Association for Computational Linguistics, 2003.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

I. O'Neill, P. Hanna, A. Yue, and W. Liu. Using probabilistic logic for dialogue strategy selection. In *Proceedings of the Paralinguistic Information and its Integration in Spoken Dialogue Systems Workshop*, pages 247–253. Springer New York, 2011.

S. Oviatt, R. Coulston, and R. Lunsford. When do we interact multimodally?: cognitive load and multimodal communication patterns. In *Proceedings of the 6th international conference on Multimodal interfaces*, pages 129–136. ACM, 2004.

T. Paek. Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment. In *Proceedings of the Interspeech Workshop "Dialogue on Dialogues - Multidisciplinary Evaluation of Advanced Speech-based Interacive Systems"*, 2006.

T. Paek and D. M. Chickering. Evaluating the markov assumption in markov decision processes for spoken dialogue management. *Language Resources and Evaluation*, 40(1):47–66, 2006.

T. Paek and R. Pieraccini. Automating spoken dialogue management design using machine learning: An industry perspective. *Speech Communications*, 50(8-9):716–729, August 2008.

C. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, August 1987.

J. S. Pardo. On phonetic convergence during conversational interaction. *The Journal of the Acoustical Society of America*, 119:2382, 2006.

R. J. Passonneau, S. L. Epstein, and T. Ligorio. Naturalistic dialogue management for noisy speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):928–942, 2012.

H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, 29:309–352, 2007.

J. Pearl. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32 (2):245–257, 1987.

O. Pietquin. Optimising spoken dialogue strategies within the reinforcement learning paradigm. In *Reinforcement Learning, Theory and Applications*, pages 239–256. I-Tech Education and Publishing, 2008.

O. Pietquin and T. Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech and Language Processing*, 14(2):589–599, December 2006.

J. Pineau. *Tractable Planning Under Uncertainty: Exploiting Structure*. PhD thesis, Robotics Institute, CA.gie Mellon University, Pittsburgh, USA, 2004a.

J. Pineau. *Tractable planning under uncertainty: exploiting structure*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2004b. AAI3155794.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 1025 – 1032, 2003.

S. Png and J. Pineau. Bayesian reinforcement learning for POMDP-based dialogue systems. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2156–2159, 2011.

S. Png, J. Pineau, and B. Chaib-draa. Building adaptive dialogue systems via bayes-adaptive POMDPs. *Journal of Selected Topics in Signal Processing*, 6(8):917–927, 2012.

M. Poesio and H. Rieser. An incremental model of anaphora and reference resolution based on resource situations. *Dialogue & Discourse*, 2(1):235–277, 2011.

J. M. Porta, N. Vlassis, M. T.J. Spaan, and P. Poupart. Point-based value iteration for continuous pomdps. *Journal of Machine Learning Research*, 7:2329–2367, December 2006.

P. Poupart. *Exploiting structure to efficiently solve large scale partially observable markov decision processes*. PhD thesis, University of Toronto, Toronto, Canada, 2005.

Pascal Poupart and Nikos A. Vlassis. Model-based bayesian reinforcement learning in partially observable domains. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2008.

P. Prodanov and A. Drygajlo. Bayesian networks for spoken dialogue management in multimodal systems of tour-guide robots. In *Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech)*, pages 1057–1060, 2003.

M. Purver. *The Theory and Use of Clarification Requests in Dialogue*. PhD thesis, King's College, University of London, August 2004.

A. Raux and M. Eskenazi. A finite-state turn-taking model for spoken dialog systems. In *HLT-NAACL*, pages 629–637. The Association for Computational Linguistics, 2009.

A. Raux and Y. Ma. Efficient probabilistic tracking of user goal and dialog history for spoken dialog systems. In *Proceedings of Interspeech 2011*, Florence, Italy, 2011.

A. Raux, B. Langner, D. Bohus, A. W. Black, and M. Eskenazi. Let's go public! taking a spoken dialog system to the real world. In *INTERSPEECH*, pages 885–888. ISCA, 2005.

M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

V. Rieser and O. Lemon. Using logistic regression to initilise reinforcement-learning-based systems. In *Proceedings of the IEEE Spoken Language Technology Workshop (SLT 2006)*, pages 190–193, 2006.

V. Rieser and O. Lemon. Natural language generation as planning under uncertainty for spoken dialogue systems. In Emiel Krahmer and Mariët Theune, editors, *Empirical methods in natural language generation*, pages 105–120. Springer-Verlag, Berlin, Heidelberg, 2010a.

V. Rieser and O. Lemon. Learning human multimodal dialogue strategies. *Natural Language Engineering*, 16:3–23, 2010b.

V. Rieser and J. D. Moore. Implications for generating clarification requests in task-oriented dialogues. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 239–246. Association for Computational Linguistics, 2005.

E. K. Ringger and J. F. Allen. Error correction via a post-processor for continuous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1996)*, pages 427–430, Washington, DC, USA, 1996. IEEE Computer Society.

S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, July 2008.

S. Ross, J. Pineau, B. Chaib-draa, and P. Kreitmann. A Bayesian Approach for Learning and Planning in Partially Observable Markov Decision Processes. *Journal of Machine Learning Research*, 12:1729–1770, 2011.

Stéphane Ross and Brahim Chaib-draa. AEMS: An anytime online search algorithm for approximate policy refinement in large POMDPs. In Manuela M. Veloso, editor, *IJCAI*, pages 2592–2598, 2007.

D. Roy. Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence*, 167(1-2):170–205, 2005.

N. Roy, J. Pineau, and S. Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 93–100. Association for Computational Linguistics, 2000.

N. Roy, G. Gordon, and S. Thrun. Finding approximate pomdp solutions through belief compression. *Journal of Artificial Intelligence Research*, 23(1):1–40, January 2005.

G. A. Rummery. *Problem Solving with Reinforcement Learning*. PhD thesis, Cambridge University, 1995.

H. Sacks, E. A. Schegloff, and G. Jefferson. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735, dec 1974.

M. Salem, S. Kopp, I. Wachsmuth, K. Rohlfing, and F. Joublin. Generation and evaluation of communicative robot gesture. *International Journal of Social Robotics, Special Issue on Expectations, Intentions, and Actions*, 2012.

M. Salem, S. Kopp, and F. Joublin. Generating finely synchronized gesture and speech for humanoid robots: a closed-loop approach. In *HRI*, pages 219–220. IEEE/ACM, 2013.

S. Sanner. Relational Dynamic Influence Diagram Language (RDDL): Language Description. 2010.

S. Sanner and K. Kersting. Symbolic dynamic programming for first-order pomdps. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*, 2010.

J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL 2007, pages 149–152, 2007a.

J. Schatzmann, B. Thomson, and S. Young. Error simulation for training statistical dialogue systems. In *ASRU*, pages 526–531. IEEE, 2007b.

K. Scheffler and S. Young. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the second international conference on Human Language Technology (HLT 2002)*, pages 12–19, San Francisco, CA, USA, 2002.

D. Schlangen. Towards finding and fixing fragments: using ml to identify non-sentential utterances and their antecedents in multi-party dialogue. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 247–254. Association for Computational Linguistics, 2005.

D. Schlangen and A. Lascarides. The interpretation of non-sentential utterances in dialogue. In *Proceedings of the 4th SIGDIAL Workshop on Discourse and Dialogue*, 2003.

D. Schlangen and G. Skantze. A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2009)*, pages 710–718. Association for Computational Linguistics, 2009.

D. Schlangen, T. Baumann, and M. Atterer. Incremental reference resolution: The task, metrics for evaluation, and a Bayesian filtering model that is sensitive to disfluencies. In *Proceedings of the 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2009)*, pages 30–37. Association for Computational Linguistics, 2009.

D. Schlangen, T. Baumann, H. Buschmeier, S. Kopp, G. Skantze, and R. Yaghoubzadeh. Middleware for incremental processing in conversational agents. In *Proceedings of the 11th Annual Meeting of the Special Interest Group in Discourse and Dialogue (SIGDIAL 2010)*, pages 51–54. Association for Computational Linguistics, 2010.

J. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969.

J. Searle. *Expression and Meaning. Studies in the Theory of Speech acts*. Cambridge University Press, 1979.

S. Seneff and J. Polifroni. Dialogue Management in the Mercury Flight Reservation System. In *ANLP-NAACL Workshop on Conversational Systems*, Seattle, 2000.

G. Shani, J. Pineau, and R. Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

E. Shriberg, A. Stolcke, D. Jurafsky, N. Coccaro, M. Meteer, R. Bates, P. Taylor, K. Ries, R. Martin, and C. Van Ess-Dykema. Can prosody aid the automatic classification of dialog acts in conversational speech? *Language and speech*, 41(3-4):443–492, 1998.

D. Silver and J. Veness. Monte-carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems (NIPS 2010)*, pages 2164–2172, 2010a.

D. Silver and J. Veness. Monte-Carlo Planning in Large POMDPs. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS 2010)*, 2010b.

B. W. Silverman. *Density estimation: for statistics and data analysis*. Chapman and Hall, 1986.

J. Sinclair and M. Coulthard. *Towards an Analysis of Discourse*. Oxford University Press, 1975.

S. P. Singh, D. J. Litman, M. J. Kearns, and M. A. Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002.

Satinder P. Singh, Michael J. Kearns, D. J. Litman, and Marilyn A. Walker. Empirical evaluation of a reinforcement learning spoken dialogue system. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 645–651. AAAI Press, 2000.

G. Skantze. *Error Handling in Spoken Dialogue Systems: Managing Uncertainty, Grounding and Miscommunication*. PhD thesis, Royal Institute of Technology (KTH), Stockholm, 2007.

E. J. Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, 1971.

O. Ståhl, B. Gambäck, M. Turunen, and J. Hakulinen. A mobile health and fitness companion demonstrator. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2009)*, pages 65–68. Association for Computational Linguistics, 2009.

M. Steedman and R. P. A. Petrick. Planning dialog actions. In *Proceedings of the 8th SIGDIAL Meeting on Discourse and Dialogue*, pages 265–272, Antwerp, Belgium, 2007.

A.J. Stent and S. Bangalore. Interaction between dialog structure and coreference resolution. In *Proceedings of the IEEE Spoken Language Technology Workshop (SLT 2010)*, pages 342–347, 2010.

R. Stiefelhagen, C. Fugen, R. Gieselmann, H. Holzapfel, K. Nickel, and A. Waibel. Natural human-robot interaction using speech, head pose and gestures. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, volume 3, pages 2422–2427. IEEE, 2004.

T. Stivers, N. J. Enfield, P Brown, C. Englert, M. Hayashi, T. Heinemann, G. Hoymann, F. Rossano, J. P de Ruiter, K.-E. Yoon, and S. C. Levinson. Universals and cultural variation in turn-taking in conversation. *Proceedings of the National Academy of Sciences*, 106(26):10587–10592, 2009.

A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Van Ess-Dykema, and M. Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373, 2000.

M. Stone, C. Doran, B. Webber, T. Bleam, and M. Palmer. Microplanning with communicative intentions: The SPUD system. *Computational Intelligence*, 19(4):311–381, 2003.

M. Strube and C. Müller. A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 168–175. Association for Computational Linguistics, 2003.

N. Ström and S. Seneff. Intelligent barge-in in conversational systems. In *Proceedings of the 6th International Conference of Spoken Language Processing (ICSLP/Interspeech 2000)*, pages 652–655. ISCA, 2000.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009)*, pages 993–1000. ACM, 2009.

J. Tetreault and D. Litman. Using reinforcement learning to build a better model of dialogue state. In *Proceedings of the 11th Conference of the European Association for Computational Linguistics (EACL 2006)*, 2006.

R. H. Thomason and M. Stone. Enlightened update: A computational architecture for presupposition and other pragmatic phenomena. In *Presupposition Accommodation*. Ohio State Pragmatics Initiative, 2006.

B. Thomson, F. Jurcícek, M. Gasic, S. Keizer, F. Mairesse, K. Yu, and S. Young. Parameter learning for POMDP spoken dialogue models. In *SLT*, pages 271–276. IEEE, 2010. ISBN 978-1-4244-7903-0.

B. Thomson, M. Gašić, M. Henderson, P. Tsiakoulis, and S. Young. N-best error simulation for training spoken dialogue systems. In *SLT*, pages 37–42. IEEE, 2012.

V. Thomson and S. Young. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24:562–588, October 2010.

M. Tomasello, M. Carpenter, J. Call, T. Behne, and H. Moll. Understanding and sharing intentions: The origins of cultural cognition. *Behavioral and Brain Sciences*, 28:675–691, 9 2005.

D. R. Traum. *A Computational Theory of Grounding in Natural Language Conversation*. PhD thesis, University of Rochester, Rochester, NY, USA, 1994.

D. R. Traum and J. F. Allen. Discourse obligations in dialogue processing. In *Proceedings of the 32nd annual meeting of the Association for Computational Linguistics*, pages 1–8. Association for Computational Linguistics, 1994.

D. R. Traum and E. A. Hinkelman. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8:575–599, 1992.

D. R. Traum, P. Aggarwal, R. Artstein, S. Foutz, J. Gerten, A. Katsamanis, A. Leuski, D. Noren, and W. R. Swartout. Ada and grace: Direct interaction with museum visitors. In *The 12th International Conference on Intelligent Virtual Agents (IVA 2012)*, volume 7502 of *Lecture Notes in Computer Science*, pages 245–251. Springer, 2012.

M. Turunen. *Jaspis–A Spoken Dialogue Architecture and Its Applications*. PhD thesis, University of Tampere, Department of Computer Sciences, Finland, 2004.

G.-J. Van Noord, G. Bouma, R. Koeling, and M.-J. Nederhof. Robust grammatical analysis for spoken dialogue systems. *Natural Language Engineering*, 5:45–93, 2 1999.

M. van Otterlo. A survey of reinforcement learning in relational domains. Technical report, University of Twente, 2006.

M. van Otterlo. Solving relational and first-order logical markov decision processes: A survey. In *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 253–292. Springer Berlin Heidelberg, 2012.

R. C. Vipperla, M. Wolters, K. Georgila, and S. Renals. Speech input from older users in smart environments: Challenges and perspectives. In *Proceedings of HCI International: Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments*, number 5615 in Lecture Notes in Computer Science. Springer, 2009.

T. Visser, D. Traum, D. DeVault, and R. op den Akker. Toward a model for incremental grounding in spoken dialogue systems. In *The 12th International Conference on Intelligent Virtual Agents (IVA 2012)*, Santa Cruz, CA, September 2012.

N. Vlassis, M. Ghavamzadeh, S. Mannor, and P. Poupart. Bayesian reinforcement learning. In M. Wiering and M. Otterlo, editors, *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 359–386. Springer, 2012.

W. E. Wahlster. *SmartKom: Foundations of Multimodal Dialogue Systems*. Cognitive Technologies. Springer Verlag, 2006.

M. Walker, B. Pellom, J. Polifroni, A. Potamianos, P. Prabhu, A. Rudnicky, S. Seneff, and D. Stallard. DARPA Communicator dialog travel planning systems: The June 2000 data collection. In *Proceedings of the 7th European Conference on Speech Communication and Technology (Interspeech 2001)*, pages 1371–1374, 2001.

M. A. Walker. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, 12(1):387–416, 2000.

C. Wang and R. Khardon. Relational partially observable MDPs. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*, pages 1153–1158, 2010.

C. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

J. D. Williams. Exploiting the ASR n-best by tracking multiple dialog state hypotheses. In *Proceedings of the 9th Annual Conference of the International Speech Communication Association (Interspeech 2008)*, pages 191–194. ISCA, 2008a.

J. D. Williams. The best of both worlds: Unifying conventional dialog systems and POMDPs. In *International Conference on Speech and Language Processing (ICSLP 2008)*, Brisbane, Australia, 2008b.

J. D. Williams. Incremental partition recombination for efficient tracking of multiple dialog states. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5382–5385, 2010.

J. D. Williams and S. Young. Using Wizard-of-Oz simulations to bootstrap Reinforcement- Learning based dialog management systems. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, 2003.

J. D. Williams and S. Young. Scaling up POMDPs for dialog management: The "summary pomdp" method. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2005)*, pages 177–182, 2005.

J. D. Williams and S. Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21:393–422, 2007.

J. D. Williams, P. Poupart, and S. Young. Factored Partially Observable Markov Decision Processes for Dialogue Management. In *Proceedings of the 4th Workshop on Knowledge and Reasoning in Practical Dialog Systems*, 2005.

J. D. Williams, P. Poupart, and S. Young. Partially Observable Markov Decision Processes with continuous observations for dialogue management. In L. Dybkjær and W. Minker, editors, *Recent Trends in Discourse and Dialogue*, volume 39 of *Text, Speech and Language Technology*, pages 191–217. Springer Netherlands, 2008.

D. Wilson and D. Sperber. Relevance theory. *Handbook of pragmatics*, 2002.

M. Wölfel and J. McDonough. *Distant speech recognition*. Wiley, 2009.

F. Xu, S. Schmeier, R. Ai, and H. Uszkoreit. Yochina: Mobile multimedia and multimodal crosslingual dialog system. In *Proceedings of International Workshop On Spoken Dialogue Systems Technology (IWSDS 2012)*. Springer, 2012.

H. L. S. Younes and M. L Littman. PPDDL1.0: The language for the probabilistic part of IPC-4. In *Proceedings of the 4th International Planning Competition*, 2004.

S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24:150–174, 2010.

S. Young, M. Gačić, B. Thomson, and J. D. Williams. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

H. Zender, G.-J. M. Kruijff, and I. Kruijff-Korbayová. Situated resolution and generation of spatial referring expressions for robotic assistants. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 1604–1609, Pasadena, CA, USA, July 2009.

B. Zhang, Q. Cai, J. Mao, E. Chang, and B. Guo. Spoken Dialogue Management as Planning and Acting under Uncertainty. In *Proceedings of 7th European Conference on Speech Communication and Technology*, pages 2169–2172, 2001.

N. Lianwen Zhang and D. Poole. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research (JAIR)*, 5:301–328, 1996.

P. Zhang, Q. Zhao, and Y. Yan. A spoken dialogue system based on keyword spotting technology. In J. A. Jacko, editor, *Proceedings of the 12th international conference on Human-Computer Interaction (HCI 2007)*, volume 4552 of *Lecture Notes in Computer Science*, pages 253–261. Springer, 2007.

R. Zhou and R. A. Hansen. An improved grid-based approximation algorithm for POMDPs. In *Proceedings of the 17th international joint conference on Artificial intelligence (IJCAI 2001)*, pages 707–714, San Francisco, CA, USA, 2001.

V. Zue, S. Seneff, J.R. Glass, J. Polifroni, C. Pao, T.J. Hazen, and L. Hetherington. JUPlTER: a telephone-based conversational interface for weather information. *IEEE Transactions on Speech and Audio Processing*, 8(1):85–96, 2000.