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

July 8, 2013

Abstract

TODO

Contents

Mathematical notations

Probability distributions:

X	Random variable
$Val(X)$	Range of values for the variable X
$P(X)$	Probability distribution for the random variable X
$P(X_1, ...X_n)$	Joint probability distribution for X_1, ... X_n
$P(X_1, ...X_n \mid Y_1, ...Y_n)$	Conditional probability distribution for X_1, ... X_n given Y_1, ... Y_n
$E(X)$	Expectation of the random variable X

Graphical models:

$(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$	Conditional independence of the 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 $\forall\, Y \in parents(X), Y \rightarrow X$
$P(\mathbf{Q} \mid \mathbf{E}=\mathbf{e})$	Probability query on variables $\mathbf{Q}$ given evidence $\mathbf{E}=\mathbf{e}$
$U(\mathbf{Q} \mid \mathbf{E}=\mathbf{e})$	Utility query on variables $\mathbf{Q}$ given evidence $\mathbf{E}=\mathbf{e}$
$\boldsymbol{\theta}_{X_i}$	Parameters associated with the random variable X_i

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 $\in \Re^{\lvert S \rvert - 1}$
$V(b)$	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
$\tilde{u}_u$	ASR recognition hypotheses for user utterance
a_u	User dialogue act
$\tilde{a}_u$	NLU hypotheses for the user dialogue act
i_u	User intention
c	Interaction context
a_m	System dialogue act
u_m	System utterance

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 range 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 illustrates 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 decide 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.

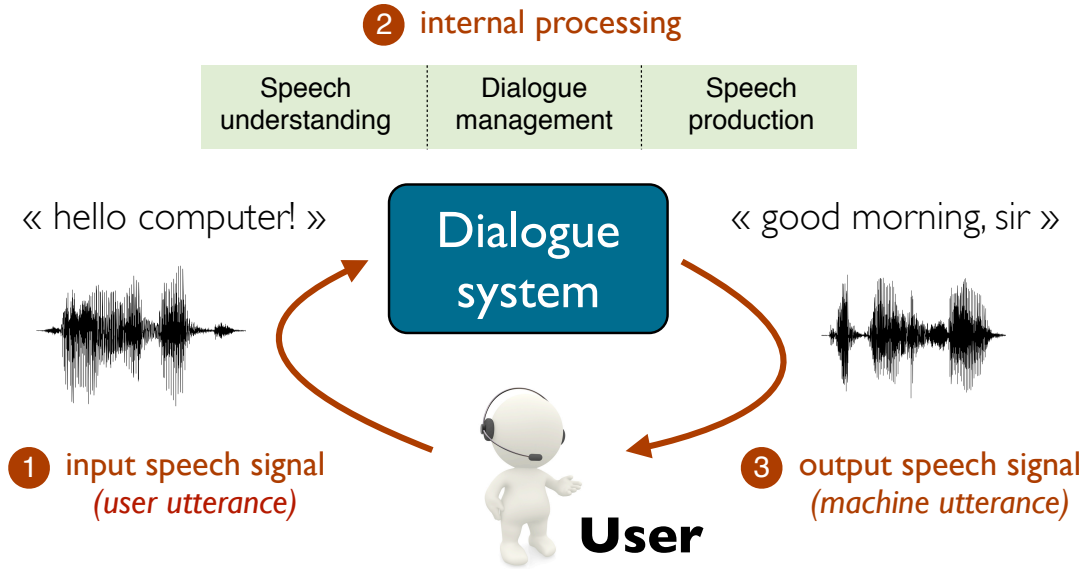


Figure 1.1: Schematic view of a spoken dialogue system.

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. Dialogue decision must also be taken under real-time conditions, 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 for computational dialogue modelling.

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.

There is an extensive body of work in the machine learning and decision-theoretic planning literature which shows how to address 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.

The theoretical underpinnings of the thesis are built upon *probabilistic graphical models* (Koller and Friedman, 2009). 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 complete 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 is 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. The relations between variables are expressed by instantiating the probabilistic rules in the graphical 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 exploitation has most often been reduced to a mere external filter to a classical model (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 different learning scenarios:

1. The first experiment, detailed in Section 5.3, focused on the problem of estimating the utilities of various system actions given a small data set collected from Wizard-of-Oz interactions.[2]

[2]A Wizard-of-Oz interaction is an experimental procedure borrowed from the field of human-computer interaction

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, 2012c). reread this part afterwards?

2. The second experiment, described in Section 6.3, 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 a software toolkit that implements all the representations and algorithms presented in this work. The toolkit is called `openDial` and is freely available under an open source licence.[3] It enables system developers to design, evaluate and deploy 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. Its implementation is described in Appendix C.

(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.

[3] The toolkit can be downloaded at `http://opendial.googlecode.com`.

We carried out all the experiments described in this thesis in a *human–robot interaction* (HRI) domain. The selection of this application domain as a test bed for our framework was motivated by two factors. First of all, HRI domains often embody a rich mix of contextual features extracted from the situated environment and the tasks to complete by the agent. Moreover, HRI domains must frequently experience significant levels of uncertainty due to imperfect sensors, unreliable motors, and failure-prone speech recognition.[4]

The Nao robot from Aldebaran Robotics was used as a platform for all our experiments.[5] An example of interaction with the robot is shown in Figure 1.2. 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 automatically perceived by the robot. A detailed description of the evaluation setups used in the experiments is provided in the Chapters 4–6.

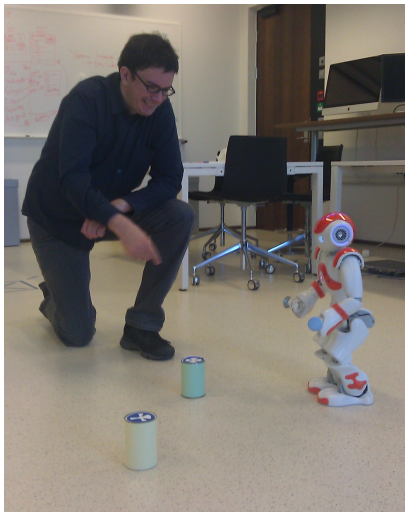


Figure 1.2: Human user interacting with the Nao robot.

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. In particular, we review both hand-crafted and statistical approaches to the design of dialogue strategies.

Chapter 3: Probabilistic Modelling of Dialogue

We start by reviewing the core notions of directed graphical models, since they constitute the formal basis for our framework. write this

[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.

Chapter 4: Probabilistic Rules

This chapter lays down the theoretical foundations of our approach. 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 models. Following this, we explain how the rules are practically instantiated in the Bayesian Network representing the dialogue state. The chapter also addresses some advanced modelling questions, and concludes by discussing related work that also aimed at reducing the dimensionality problem when learning dialogue strategies.

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 to estimate these parameters is grounded in Bayesian inference. To validate our approach, we detail an experiment demonstrating how to learn the utilities of a set of actions from Wizard-of-Oz data collected in a human–robot interaction domain. The experiment illustrates in particular the benefits of applying probabilistic rules.

Chapter 6: Learning from Interactions

This chapter builds upon the previous chapter and extends it to a reinforcement learning context. We show that is is possible to efficiently learn the parameters of dialogue models 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 in a HRI domain, and evaluated the relative performance of a model structured with probabilistic rules compared to a plain probabilistic model.

Chapter 7: User Evaluation

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

Chapter 8: 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 terms). 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 govern 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 should therefore best be 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 their discussion 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?*"). This common ground grows 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 rigorously 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 previous chapter, Figure 1.1 represented a dialogue system as a black box taking speech inputs from the user and generating spoken responses. Real 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. It should be noted that 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, 2009; Schlangen et al., 2010).

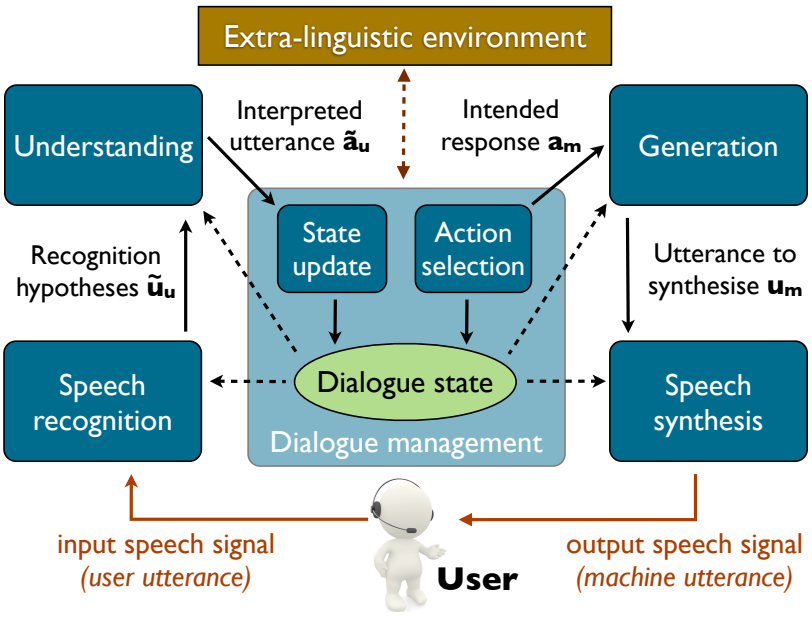


Figure 2.1: Information flow for a typical spoken dialogue system. The solid lines denote necessary input and outputs while the dotted lines represent optional contextual information.

Spoken dialogue systems can rely on 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 eg. 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 exploited 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 (Goodrich and Schultz, 2007), a virtual world for embodied virtual agents (Kopp et al., 2003), 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; 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, has 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 allow for incremental processing at various stages of interpretation and decision-making (Schlangen and Skantze, 2009).

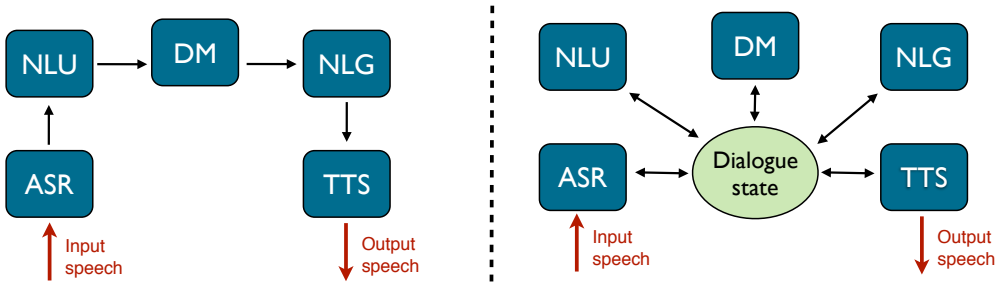


Figure 2.2: Comparison between pipeline (left) and ISU (right) system architectures. Abbreviations: ASR = *Automatic Speech Recognition*, NLU = *Natural Language Understanding*, DM = *Dialogue Management*, NLG = *Natural Language Generation*, and TTS = *Text-to-Speech Synthesis*.

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 formalisation rests on the representation 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, 2010).

[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 recogniser 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. This dialogue state should encode every information that is of general relevance for the dialogue system, such as the recent dialogue history (encoded as a temporally ordered sequence of dialogue acts performed by the conversational partners), 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 provided by other modules. 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 define features that attempt to characterise the user and her/his preferences. Depending on the theoretical premises chosen by the system designer, the dialogue state can either be encoded as a fully observable data structure, or can be extended to explicitly represent partial observability through the definition of probability distributions on the values of the state variables.

The second task of dialogue management is to make decisions based on this dialogue state. This task is often called *action selection*. The decision pertains to the next action to perform by the system, and 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 process can take many forms, from the application of rules to the use of offline or online planning techniques.

The dialogue management step 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 encoded in a logical form with predicates and arguments.

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

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 techniques can be adopted, 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 the case of multimodal systems, the module might 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 mapping is performed in two consecutive stages. The utterance is first converted into a phonemic representation. This conversion involves various operations related to text normalisation, phonetic and prosodic analysis. Once this conversion is completed, the resulting phonemic representation is fed into 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 where voice interfaces are to be preferred for safety reasons (Hansen et al., 2005; Castronovo et al., 2010). 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 applied to 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 settings (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, 2008), 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 relies 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 update the current state from the edge source to its target. Figure 2.3 illustrates an example of FSA 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 legal input is provided. The system will continue to request directions until the "*stop*" command is uttered.

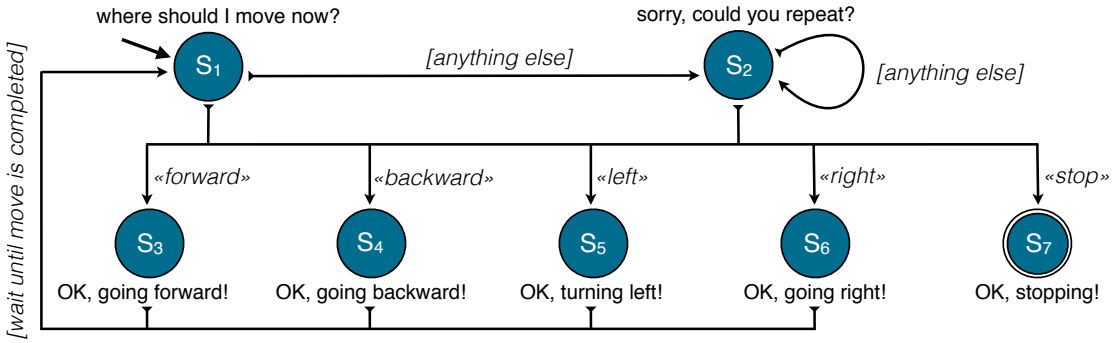


Figure 2.3: Example of finite-state automaton (FSA) for dialogue management, with seven possible states. The starting state for this FSA is s_1 and the (unique) ending state is s_7. The edges $s_3, ...s_7 \rightarrow s_1$ are traversed once the movement is completed, and the two edges $s_1, s_2 \rightarrow s_2$ 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 provides a simple and efficient way to design a dialogue manager. Their expressive power is however 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. Finite-state automata are therefore difficult to scale to larger 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

To overcome the rigidity of finite-state automata, richer representations of the dialogue state are required. A popular solution is apply frame-based 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 mark 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 used 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 based on communicative principles of rational and coordinated interaction between agents.

Benefits and limitations of hand-crafted approaches

The hand-crafted approaches to dialogue management we have described are attractive due to their ability to capture rich conversational phenomena. 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 notable shortcomings:

1. They assume complete observability of the dialogue context and provide only a very 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 the 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 category of learning algorithm, the representation of the dialogue state and policy, and the type of 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. Dialogue is fundamentally 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 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 is 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 systems, 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 not directly observable anymore. 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., 2012).

The next chapter provides more details on the theoretical foundations of these modelling strategies and their applications to practical 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 possibility 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 appropriate 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 they 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 to address this issue (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 deal with even richer state-action spaces, with multiple tasks to perform, sophisticated user models, and a complex, dynamic context. 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. Due to the automatic optimisation mechanism, incorporating such general constraints in a statistically learned dialogue policy can be a complex procedure. 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 (Williams, 2008b; Henderson et al., 2008), 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 present a comparison of the most important hand-crafted and statistical methods to dialogue management in terms of state representation, account of uncertainty, 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 management. Starting with a linguistic analysis of the most important dialogue 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 practically designed. 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 environmen-

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 or 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 or 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 or factored state	yes	Update of belief state	Policy optimised via reinforcement learning from real or simulated dialogues
Approach presented in this thesis	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.

tal factors. The range of possible applications of dialogue system technology is particularly 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 social 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 whose role is to encode the system knowledge of the current conversational situation. This dialogue state can vary greatly in complexity depending on the chosen framework – from the atomic symbols used in finite-state approaches to the rich nested feature structures found 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. This optimisation can be performed using various learning techniques, from supervised learning on a Wizard-of-Oz data set to reinforcement learning with a user simulator and a general 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.

The last section concluded its 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 modelling of dialogue

The previous chapter provided an short overview of the most important 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 dialogue management.

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 powerful, 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: 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 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 employed to model dialogue management tasks. The section reviews the wide range of 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 exist 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 ordered set of random variables, where each variable X_i is associated with a range of mutually exclusive values. This range can be either discrete (finite or infinite) 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 contain 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 $parents(X_i)$. Note that this set 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 (CPDs) fully determine the joint probability distribution of the Bayesian network.

Given such definition, 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 variables 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.

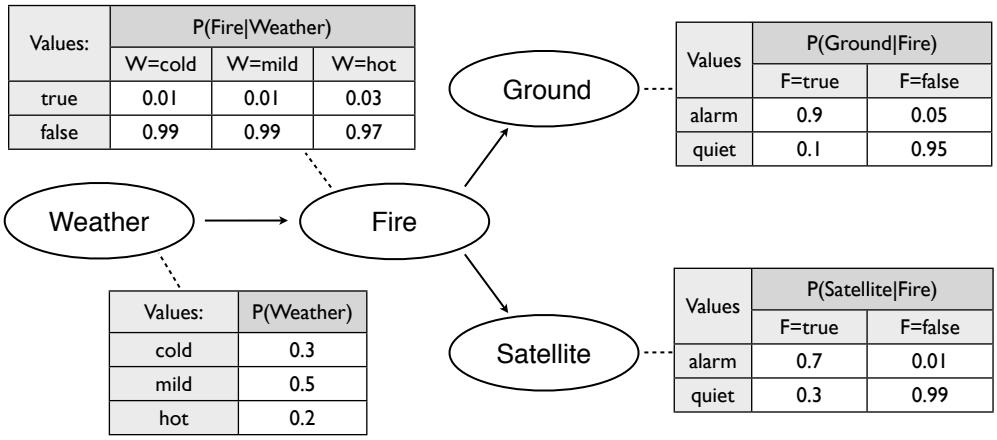


Figure 3.1: Example of Bayesian network with four random variables.

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

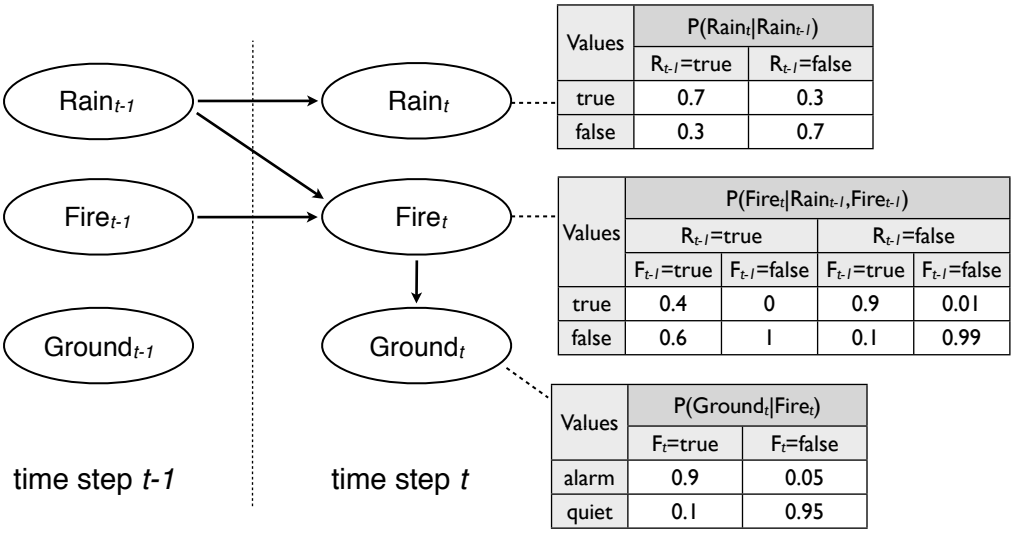
In order to apply Bayesian networks to tasks such as dialogue management, two additional elements are necessary. The first extension is to allow variables to evolve as a function of time. Such temporal dependencies are indeed necessary to account for the dynamic nature of dialogue (the dialogue state is not a static entity and 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. Graphical models for decision-making must therefore explicitly represent the relation between state variables, actions, and utilities.

Dynamic decision networks[4] (DDNs) extend the dynamic Bayesian networks described in the previous section with a representation of action variables and their corresponding utilities. Dynamic decision networks may include three classes of nodes:

1. *Chance nodes* correspond to the classical random variables that have been discussed so far. As for the previous types of graphical models, 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.

Figure 3.3 illustrates an example of dynamic decision network with a decision variable with two alternative actions and a utility variable encoding its utility depending on the occurrence of a fire.

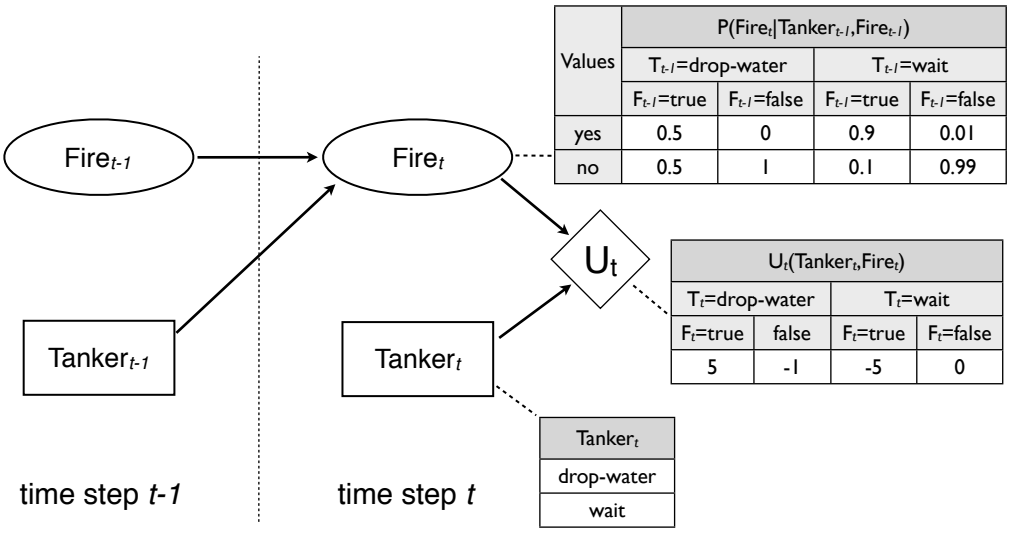


Figure 3.3: Example of dynamic decision network. By convention, chance nodes are represented with circles, decision nodes with squares, and utility nodes with diamonds.

[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 and $\mathbf{E} \subset \mathbf{X}$ 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 for exact inference can be alternatively used, such as message passing on clique trees (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 and a specific type of importance sampling algorithm called likelihood weighting (see below). These algorithms are used 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. Appendix C provides more detail on the technical aspects of this implementation.

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 many of the experiments conducted in this thesis.

We limit the present discussion to inference in Bayesian networks, but the sampling algorithm can be extended to decision networks in a straightforward manner.

The intuition behind all 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 $P(\mathbf{Q} \mid \mathbf{E} = \mathbf{e})$, every sample is associated with a specific *weight* that expresses the likelihood of the evidence given the assignment for all the other variables. The pseudocode in Algorithm 1 (modified from Russell and Norvig (2010); Koller and Friedman (2009)) outline the procedure. The notation $\mathbf{x}(\mathbf{Y})$ is used to refer to the values specified for the variables $\mathbf{Y}$ in the assignment $\mathbf{x}$.

Algorithm 1 LIKELIHOOD-WEIGHTING $(\mathcal{B}, \mathbf{Q}, \mathbf{E} = \mathbf{e}, N)$

Input: Bayesian 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 W be a vector of weighted counts for each possible value for Q, initialised to zero
for i = 1 → N do
    Initialise sample x ← ⟨e⟩ and weight w ← 1
    for all X_i ∈ X_1, ...X_n do
        if X_i ∈ E then
            w ← w × P(X_i = e(X_i) | x)
        else
            x_i ← random sample drawn from P(X_i | x)
            x ← x ∪ ⟨x_i⟩
        end if
    end for
    W[x(Q)] ← W[q] + w
end for
Normalise the counts in W
return W
```

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, 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.

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

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 rely 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 is required to simultaneously 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.

For dialogue management, the graph structure of the dialogue models can often be designed by the application designer, and we shall therefore concentrate on the parameter estimation problem.

Maximum likelihood estimation

The most straightforward 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 data set. Given a data set $\mathbf{d}$, 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 likelihood $P(\mathbf{d} \mid \boldsymbol{\theta})$, often written in logarithmic form:

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} P(\mathbf{d} \mid \boldsymbol{\theta}) = \operatorname*{argmax}_{\boldsymbol{\theta}} \log P(\mathbf{d} \mid \boldsymbol{\theta}) \tag{3.5}$$

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 and the MLE estimate is not generally amenable to a closed-form solution. Optimisation methods are thus required to find the optimal parameter values. Two common methods for this optimisation are 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 may lead to very 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 $P(X_i \mid parents(X_i))$ with unknown parameters is therefore associated with a parent node $\theta_{X_i|parents(X_i)}$ that define its parameter distribution. This parameter distribution is often continuous and multivariate. Intuitively, we can think of the variable $\theta_{X_i|parents(X_i)}$ as 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 $\mathbf{d}$, the posterior distribution $P(\boldsymbol{\theta} \mid \mathbf{d})$ is given by Bayes' rule:

$$P(\boldsymbol{\theta} \mid \mathbf{d}) = \frac{P(\mathbf{d} \mid \boldsymbol{\theta})\, P(\boldsymbol{\theta})}{P(\mathbf{d})} \tag{3.6}$$

The posterior distribution $P(\boldsymbol{\theta} \mid \mathbf{d})$ 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 $P(\boldsymbol{\theta} \mid d)$ after observing a data point 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 $\theta_{Weather}$ are three-dimensional vectors $\langle \theta^1_{Weather}, \theta^2_{Weather}, \theta^3_{Weather} \rangle$, with the standard constraints on probability values: $\theta^i_{Weather} \geq 0$ for $i = \{1, 2, 3\}$ and $\theta^1_{Weather} + \theta^2_{Weather} + \theta^3_{Weather} = 1$. 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):

$$P(\mathbf{d} \mid \boldsymbol{\theta}) = \prod_{\theta_i \in \boldsymbol{\theta}} P(\mathbf{d} \mid \theta_i) \tag{3.7}$$

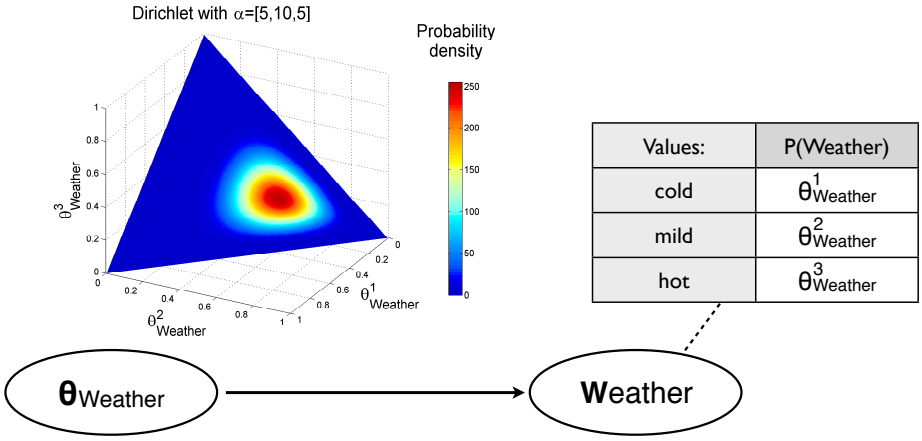


Figure 3.4: Bayesian network with variable $Weather$ and associated parameter $\theta_{Weather}$. As $Weather$ is a categorical distribution, the distribution $P(\theta_{Weather})$ is expressed as a Dirichlet with three dimensions that reflect the relative probabilities for the three values in $Val(Weather)$.

As in the maximum likelihood estimation case, the learning task becomes more complicated when dealing with partially observed data, as we can no longer represent the posterior distribution as a product of independent posteriors over each parameter. In this setting, the full posterior is often too complex (and sometimes even multimodal) 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.

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 envi-

ronment 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, 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.

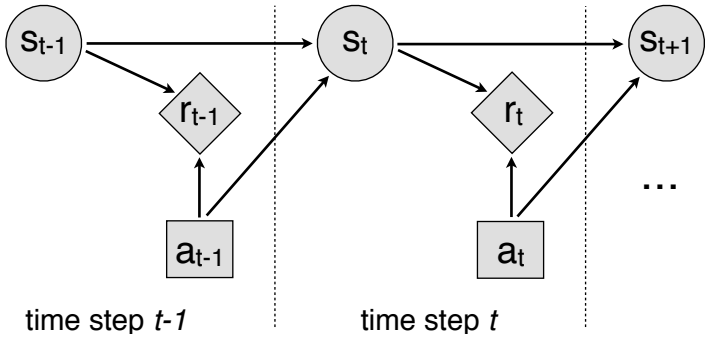


Figure 3.5: Graphical model of a Markov Decision Process (MDP) unfolded on a few time steps. Greyed entities indicate observed variables (in the MDP case, all variables are observed).

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 $V^\pi(s) = \max_a Q^\pi(s, a)$.

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:

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

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 not directly observable anymore. 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.

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 incorporate the new information. This belief update is a simple application of Bayes' theorem:

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

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

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

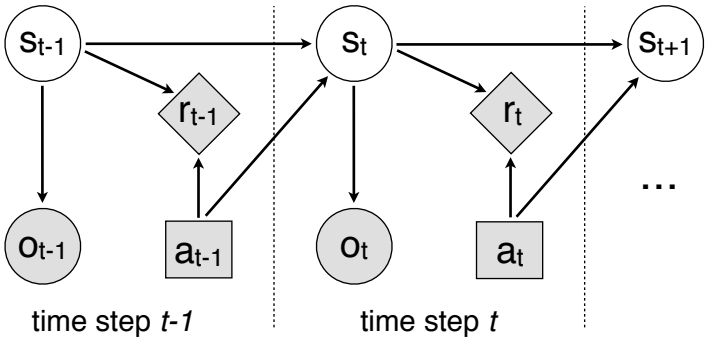


Figure 3.6: Graphical model of a Partially Observable Markov Decision Process (POMDP) unfolded on a few time steps. Compared to Figure 3.5, we notice that the state is not directly accessible anymore, but must be inferred from the observations and state history.

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 significantly 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 $a(i) \in \mathcal{A}$.[6] 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 a computationally difficult task 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 solution methods 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

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

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 particular settings, it essentially rests on the assumption that the state uncertainty will disappear after one action. It serves therefore as a poor model for information-gathering actions – that is, actions that do not change the actual state but might help in reducing state uncertainty.[7]

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 values 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). Yet another class of 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 is to rely on online planning algorithms (Ross et al., 2008; Silver and Veness, 2010). 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 planning 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 environment or task models, and be used to simultaneously learn the domain models, as demonstrated by Ross et al. (2011). The available planning time is however more limited, since planning is in this case interleaved with the system execution and must therefore meet real-time constraints.

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, 2004; 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 are however difficult to scale to large domains due to the combinatorial explosion of the state-action spaces. It is often more efficient to rely on *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 $\mathbf{O}_t$

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

connected to the state variables $\mathbf{S}_t$ through conditional dependencies $P(\mathbf{O}_t \mid \mathbf{S}_t)$ (Poupart, 2005). 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.

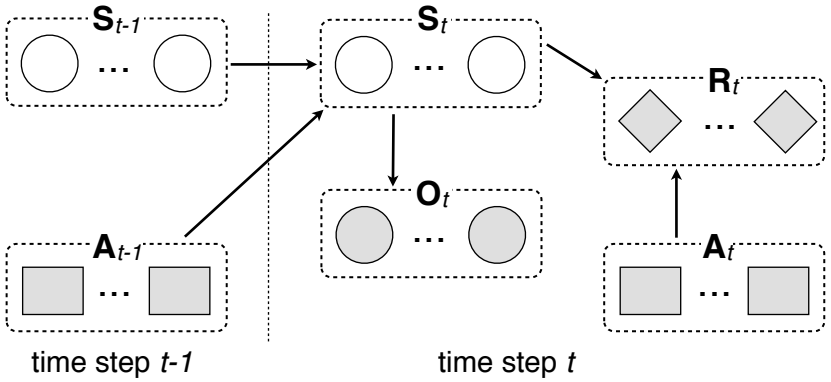


Figure 3.7: Factored representation of a POMDP with state variables $\mathbf{S}$, action variables $\mathbf{A}$, observations variables $\mathbf{O}$, and reward variables $\mathbf{R}$.

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 translated to the dialogue management task. After a brief description 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 $\{\langle s_i, a_i \rangle : 1 \leq i \leq n\}$ 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 "correct" 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).

The main challenge of supervised learning approaches is data sparsity, as only a fraction of the possible states can realistically be covered by the Wizard-of-Oz interactions. Multiple generalisation

techniques can be employed to address 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). 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 Reinforcement learning for dialogue MDPs

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.

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 therefore factored into a number of independent variables (one for each feature). Most early approaches adopted simple state representations including only essential information such as 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) include features that capture 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).

In contract to the above approaches, Henderson et al. (2008) relied on a much larger state space 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 allows 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

For most dialogue domains, the reward is fixed in advance by the system designer and reflects the task objectives. It can also be 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.[8]

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 seek to 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 learning an 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 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).[9] 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 not as valuable as real interactions, and offer no guarantee of reflecting 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 criticised (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

[8] 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).

[9] 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).

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 through the 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 intentional structures (Grosz and Sidner, 1986; Litman and Allen, 1987).

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 Reinforcement learning for dialogue POMDPs

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:

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

Belief update thus 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 and dialogue history. 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:

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

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

1. The distribution $P(a'_u \mid i'_u, a_m)$ 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 $P(i'_u \mid i_u, a_m, c')$ 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 with the parameter estimation techniques outlined in Section 3.1.3. A graphical illustration of this state factorisation is shown in Figure 3.8. 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 observation model $P(o \mid a'_u)$ 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:

$$P(o \mid a'_u) = \frac{P(a'_u \mid o)\ P(o)}{P(a'_u)} \approx P(a'_u \mid o) \tag{3.23}$$

Since the probabilities $P(a'_u \mid o)$ are provided at runtime by the ASR and NLU modules, there is no explicit specification of the observation set $\mathcal{O}$ and observation model Z at "design time" in this approximation. 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

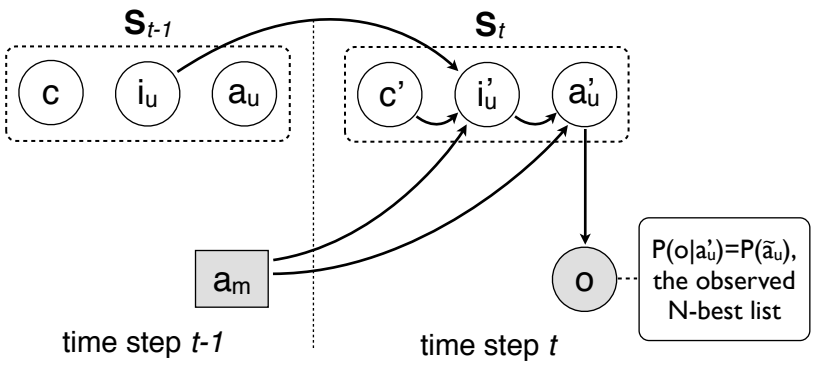


Figure 3.8: Common factorisation of the state space for a POMDP-based dialogue system, where c represents the dialogue context, i_u the user intention(s), a_u the user dialogue act, a_m the system act, and o the observed N-best list hypotheses.

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 POMDPs 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) relied on 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 verbal and non-verbal sources.

Policy optimisation

POMDP solutions methods can in theory be applied to extract the dialogue policy from a 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 derivation of a dialogue policy from interactions with a user simulator via reinforcement learning (Young et al., 2010; Thomson and Young, 2010; Daubigney et al., 2012).

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., 2012). Finally, non-parametric methods based on Gaussian Processes have recently been proposed (Gašić et al., 2011).

Model-based reinforcement learning is drastically more difficult for POMDPs than for MDPs since the dialogue state is hidden. 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) present 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) demonstrate an alternative approach 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 laid 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, and the practical exploitation of these techniques to dialogue management.

The first section of this chapter focused on the representation of probability and utility models that are able to take full advantage of the domain internal structure. We described how directed graphical models could efficiently capture (and manipulate) various probability and utility distributions. Jordan (1998, p. 1) gives convincing arguments for the use of graphical models to represent complex stochastic phenomena:

> "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. Fundamen-

> tal 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."

We detailed in particular the representational 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 second section exposed 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 learning 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 in multiple 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. We 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 important question.

Chapter 4

Probabilistic rules

This chapter spells out the dialogue modelling approach developed in this thesis. The previous chapter described how dialogue could be modelled in a probabilistic manner, and demonstrated how graphical models can help reduce the complexity of probability and utility models by exploiting independence properties between variables. But despite these attractive properties, graphical models still suffer from severe scalability problems when faced with complex dialogue domains. Dependent variables can lead to an rapid increase in the number of parameters associated with the model. Alas, only limited amounts of training data are available for most dialogue domains. Estimating the model parameters in such setting is therefore a particularly challenging task.

To address this issue, we introduce in this chapter the notion of *probabilistic rules*, which are structured mappings between conditions and (parametrised) effects. These rules function as *high-level templates* for the construction of a dynamic decision network. The key advantage of such 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 used in dialogue management, where substantial amounts of expert knowledge can be leveraged to structure the relationships between variables.

The chapter is divided in six sections: Section 4.1 demonstrates in general terms how structural assumptions can be used to refine the representation of probabilistic models. Section 4.2 show such structural assumptions can be practically encoded with probabilistic rules. We formally define these rules in terms of conditions and effects and provide some concrete examples of rules for dialogue management. Section 4.3 connects these definitions to the graphical models described in the previous chapter by showing how probabilistic rules are practically instantiated into a graphical model. Finally, Section 4.5 addresses some advanced modelling issues 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: 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 in exploiting additional structural properties.

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) 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. Theses variables are usually never observed directly but contribute to structuring the model.[1] In the particular case of medical diagnosis, the relation between predisposing factors and observed symptoms are for instance preferably described by postulating an intermediary layer of variables – possible diseases – that mediate 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.

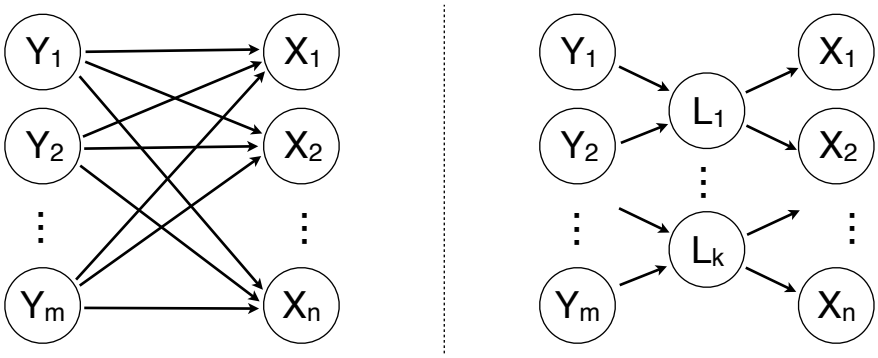


Figure 4.1: Comparison between a model that directly maps variables $\mathbf{Y}$ to $\mathbf{X}$ (left side) and one relying on latent variables $\mathbf{L}$ to serve as intermediaries (right side).

Dialogue models can often benefit from introduction of such latent variables. The transition function can in particular be modelled in terms of a limited number of latent variables, each responsible for capturing specific aspects of the interaction dynamics.

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 than on the full enumeration of combined values for the parent variables. Partitioning is an important abstraction mechanism to reduce the complexity of the model distributions and hence improve their ability to generalise to unseen examples. Utility distributions can also partition the values of their dependent variables in a similar way.

Figure 4.2 provides an example of such partitioning for the conditional probability distribution $P(Fire \mid Weather, Rain)$. The space of possible values for the parent variables is defined in this

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

example as $Val(Weather) \times Val(Rain)$ and contains 6 possible elements. We can observe that this space can be defined 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 is worth noting that the action of grouping value assignments into partitions is a modelling choice, and can degrade the model accuracy if the partitions do not reflect actual similarities in the predicted outcomes.

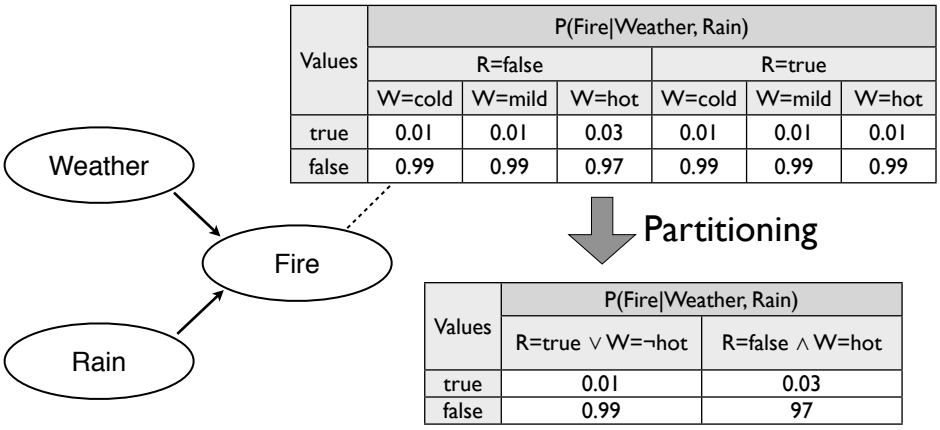


Figure 4.2: Partitioning for the conditional probability distribution $P(Fire \mid Weather, Rain)$.

Partitions must 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 will then be grouped in a partition if it satisfies the condition associated with it.

Quantification

May domains present a rich relational structure of objects or abstract entities (van Otterlo, 2006; Getoor et al., 2007). Situated dialogue domains such as human-robot interaction belong in particular to this category. 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 such relational structure 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.

First-order logic provides an excellent basis for representing and manipulating such relational structures, as it offers a rich 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 can represent such relational domains by instantiating one random variable for every possible grounding of the functions and predicates for the domain.[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 $leftOf(o_2, o_2)$. However, the definition of probability and utility distributions that can handle the relational semantics of such representation is not straightforward. Many properties or constraints are indeed independent of the particular object being considered, such as $\forall x, \neg leftOf(x, x)$ and $\forall x, y, z, leftOf(x, y) \wedge leftOf(y, z) \Rightarrow leftOf(x, z)$. Classical probabilistic models offer unfortunately no direct support for quantifiers, as their expressive power is 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). Many of the methods developed in this field operate by defining a logic-based description language on top of classical probabilistic models that allow for some form of quantification. This description language is then used as a template to generate a classical probabilistic model given a set of constants. The introduction of quantifiers provide another abstraction mechanism to reduce the complexity of a given probabilistic models by describing constraints or relations that hold for all possible groundings of a given formula and can therefore apply to a large set of random variables.

4.2 Formalisation

We now outline a generic description framework for expressing the types of internal structure we have just detailed. This description framework is based on the use of rules to structure conditional probability distributions and utility distributions. The framework has originally been presented in Lison (2012c,a).

The key idea is to represent distributions with the help of *if ... then ... else* control blocks, 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* case inside the block specifies both a *condition* on particular state variables, and a distribution over possible *effects*. The *if ... then ... else* structure is read in sequential order (as in most programming languages) until a satisfied condition is found, in which case the probabilistic

[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 equivalent to *propositionalisation* in the terminology of first-order logic.

effects associated with it are enacted.

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

4.2.1 Basic definition

Probabilistic 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 $(c_i, P(E_i))$ where 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. The probabilities $p_i^{1...m_i}$ must satisfy the usual probability axioms $p_i^j \geq 0 \; \forall i, j$ and $\sum_{j=1}^{m} p_i^j = 1 \; \forall i$.

Given these elements, a basic probabilistic 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 write $P(e_i^j)$ as a notational convenience for $P(E_i = e_i^j)$.

Conditions

The conditions c_i are expressed as formulae grounded in a subset of the state variables. This subset of state variables are the *input variables* of the rule. Conditions can be arbitrarily complex logical formulae connected by conjunction, disjunction and negation, and can also include free variables. The examples of partition $(Rain = true \vee Weather \neq hot)$ 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 effect will be enacted

for a given assignment of input values). 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** statement is explicitly put at the end of a rule, the framework inserts a final empty (i.e. always satisfied) condition associated with a void effect to ensure that the partitioning is exhaustive.

The conditions on the input variables effectively provides a compact partitioning of the state space to mitigate the dimensionality curse. Without this partitioning in alternative conditions, a rule ranging over v input variables each of size w would need to enumerate v^w possible assignments. Partitioning this space with conditions reduces this number to n mutually exclusive partitions, where n is usually small.

Effects

Associated to each condition c_i stands a list of alternative, mutually exclusive effects $e_i^{1...m_i}$. Each effect e_i^j defines a specific assignment of values for another subset of the state variables. An effect is defined as a conjunction of (variable,value) pairs $X_1 = x_1 \wedge X_2 = x_2 \wedge ... X_k = x_k$ where $X_1, ... X_k$ denote particular random variables in the dialogue state and $x_1, ... x_k$ corresponding values for these variables. The variables $X_1, ..., X_k$ are the *output variables* of the rules. In the partitioning example from Figure 4.2, the output variable is unique and corresponds to $Fire$. We shall however encounter some examples of rules with more than one output variable. It is also worth nothing that the sets of input and output variables can overlap.

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

Probabilities

Each effect e_i^j is assigned with a probability $p_i^j = P(E_i = e_i^j)$. These probabilities can be either fixed by hand or correspond to parameters to estimate from data. In the latter case, we adopt a Bayesian learning approach (cf. Section 3.1.3) and assume the probabilities $e_i^{1...m_i}$ to be drawn from the conjugate prior of categorical distributions: Dirichlet distributions. Chapters 5 and 6 detail how Dirichlet distributions can be exploited to estimate probability parameters.

It often happens that only the distribution $P(E_i)$ only includes one single effect with probability 1. In such case, the rule is equivalent to a deterministic distribution.

Example

Rule r_1 illustrates a simple example of probabilistic 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 reliance on high-level conditions enables the conditional probability distribution to be specified with only four probabilities in comparison to 12 for the original conditional probability distribution shown in Figure 4.2.

4.2.2 Utility distributions

The rule-based formalism we have outlined can also be used to described utility distributions using only minor notational changes. Utility rules essentially retain the same form as probabilistic rules, with one notable 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 $(c_i, U(D_i))$ where c_i is a condition and $U(D_i)$ an associated utility distribution over possible assignments of decision variables. The utility distribution $U(D_i)$ specifies a set of possible decisions $Val(D_i) = \{d_i^1, ...d_i^{m_i}\}$. Each decision $d_i^j \in Val(D_i)$ has a particular utility value denoted u_i^j. Utility rules can be expressed in the following manner:

$$\begin{array}{l} \textbf{if } (c_1) \textbf{ then} \\ \quad \begin{cases} U(D_1 = d_1^1) = u_1^1 \\ \ldots \\ U(D_1 = d_1^{m_1}) = u_1^{m_1} \end{cases} \\ \ldots \\ \textbf{else} \\ \quad \begin{cases} U(D_n = d_n^1) = u_n^1, \\ \ldots \\ U(D_n = d_n^{m_n}) = u_n^{m_n} \end{cases} \end{array} \tag{4.2}$$

Informally, a utility rule stipulates the utility values associated with particular system decisions depending on conditions on the state variables. As for probabilistic rules, the conditions c_i are defined as arbitrary logical formulae on a subset of state variables. The decisions d_i^j are assignments $X_1 = x_1 \wedge X_2 = x_2 \wedge ... \wedge X_n = x_n$ where the variables $X_1, ..X_n$ are decision variables and $x_1, ...x_n$ possible values for these variables. The utility values u_i^j are real numbers.

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 frequently 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:

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

Rule r_2 stipulates that the respective utility of the two action values for the decision variable $Tanker$ depending on a condition on the state variable $Fire$.

4.2.3 Quantification

Quantifiers are an important mechanism to abstract over particular relational aspects of the domain structure. 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 free variables $x_1, x_2, ...x_k$ in its conditions and/or effects is therefore written as:

$$
\begin{aligned}
&\forall \mathbf{x} = x_1, x_2, ...x_k : \\
&\qquad \textbf{if } (c_1(\mathbf{x})) \textbf{ then} \\
&\qquad\quad \begin{cases} P(E_1 = e_1^1(\mathbf{x})) = p_1^1 \\ ... \\ P(E_1 = e_1^{m_1}(\mathbf{x})) = p_1^{m_1} \end{cases} \\
&\qquad ... \\
&\qquad \textbf{else} \\
&\qquad\quad \begin{cases} P(E_n = e_n^1(\mathbf{x})) = p_n^1, \\ ... \\ P(E_n = e_n^{m_n}(\mathbf{x})) = p_n^{m_n} \end{cases}
\end{aligned}
\tag{4.3}
$$

The formalisation allows certain variables $x_1, ...x_k$ in the conditions and effects to be *underspecified*. The mapping between conditions and effects specified by the rule is therefore instantiated for every possible assignment of the underspecified variables. The quantification mechanism introduced in (4.3) enables probabilistic rules to 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 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 $x_1, ...x_k$. Quantification can also be used for utility rules in the same manner.

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

Example

Rule r_3 is an example of rule that employs quantification on two variables denoted o and c.[5]

$$
\begin{aligned}
r_3 : \forall o, c : \\
\quad \textbf{if } (a_m = \mathit{WhatIsColour}(o) \wedge \mathit{colour}(o) = c) \textbf{ then} \\
\quad\quad \Big\{ P(a'_u = \mathit{Assert}(\mathit{HasProperty}(o, c))) = 0.95
\end{aligned}
$$

Rule r_3 is used to predict the next user dialogue act a'_u after a system-initiated question such as "What colour is the object" depending on the colour of the object that is referred to. The rule assumes the presence in the dialogue state of a random variable $\mathit{colour}(o_i)$ for each object o_i perceived by the system. The rule specifies that the user is likely (with probability 0.95) to utter a dialogue act such as "the object is X" at the next turn, where X is the actual colour of the object. Otherwise no prediction is made. If the dialogue state contains e.g. one object o_1 with $P(\mathit{colour}(o_1) = \mathit{blue}) = 0.8$ and the last system action was $\mathit{WhatIsColour}(o_1)$, the probability of the user uttering "the object is blue" is therefore 0.76. It is worth observing in this example that the variables o and c are allowed to occur both inside the name of a random variable, as in $\mathit{colour}(o)$, and inside the value of a random variable, as in $\mathit{WhatIsColour}(o)$, c and $\mathit{Assert}(\mathit{HasProperty}(o, c))$.

4.3 Rule instantiation

The rules are instantiated at runtime in the dialogue state by creating a distinct chance node for every rule. These rule nodes are essentially latent variables that serve as intermediaries between input and output variables. Albeit the occurrence 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.

The dialogue state is in our approach represented as a Bayesian network of state variables, in line with other dialogue management approaches such as Thomson and Young (2010); Bui et al. (2010). Rules are then applied upon this dialogue state. Probabilistic rules are employed to create or update particular state variables, whereas utility rules define decision variables and their corresponding utility values. The instantiated rules are equivalent to a dynamic decision network (cf. previous chapter).

We describe below the instantiation procedure for each type of rule. For simplicity's sake, we shall first limit our discussion to rules without quantifiers, and then show 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 chance node is created. This chance node represents a random variable 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),

[5] Note the absence of an explicit final *else* branch, which is then by default associated with an empty effect.

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 an artificial example. The two rules r_4 and r_5 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 random variable represented by the node r_4 has three possible values, reflecting the effects described in the rule: $Val(r_4) = \{[A = a_1], [A = a_2], \{\cdot\}\}$. Note that $\{\cdot\}$ denotes the empty effect. Similarly, the random variable r_5 has four alternative effects: $Val(r_5) = \{[A = a_2 \wedge E = e_2], [A = a_2 \wedge E = e_1], [E = e_2], \{\cdot\}\}$.

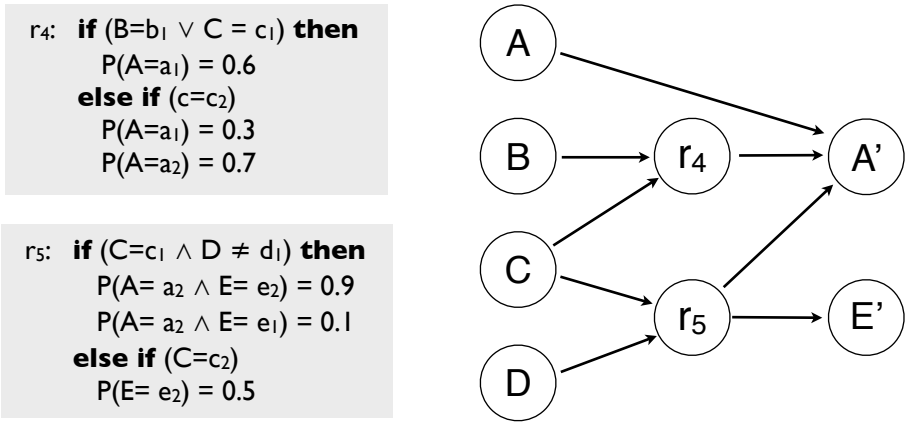


Figure 4.3: Example of instantiation for the two probability rules r_4 and r_5. The remaining probability mass in the rule specifications is by default assigned to the empty effect.

We shall adopt the following terminology for the distributions created through the instantiation procedure:

- The conditional probability distribution associated with rule nodes such as r_4 and r_5 given their inputs is a *rule distribution*.
- The conditional probability distribution associated with output variables such A' and E' given the rule nodes that affect it is a *output distribution*.

Rule distributions

The conditional probability distribution of a rule r given its input variables $I_1, ...I_n$ is defined as:

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

In other words, the rule conditions are checked in sequential order until one condition c_i is satisfied for the given input. The distribution for the rule is then simply determined as the effect distribution $P(E_i = e)$ associated with the satisfied condition c_i.

As an example, the rule distribution $P(r_4 \mid B = b_1, C = c_1)$ for the node r_4 in Figure 4.3 is defined as:

- $P(r_4 = \{A{=}a_1\} \mid B{=}b_1, C{=}c_1) = 0.6$
- $P(r_4 = \{\cdot\} \mid B{=}b_1, C{=}c_1) = 0.4$
- $P(r_4 = \{A{=}a_2\} \mid B{=}b_1, C{=}c_1) = 0$

Similarly, the distribution $P(r_5 \mid C = c_1, D = d_1)$ is a deterministic distribution with the empty effect $\{\cdot\}$ assigned to a probability 1.

Output distributions

An output variable X' is conditionally dependent on all the rule nodes that refer to it in their effects. In addition, output variables that result in the update of existing variables also include a conditional dependence on these existing variables. The output distribution is a direct reflection of the effects specified in the 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:

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

where the following notation is used:

- $\mathbf{e}$ is the conjunction of all effects, i.e. $\mathbf{e} = e_1 \wedge ...e_n$. Note that this conjunction can include several value assignments for a given variable.
- $\mathbf{e}(X)$ denotes the (possibly empty) set of values specified for the variable X in $\mathbf{e}$.
- $\mathbf{1}(p)$ is the indicator function for the Boolean p, with $\mathbf{1}(p) = 1$ if p 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 all effects $e_1, ...e_n$ are empty assignments, the value for X' is set to a default $None$ value.

In case the variable X' is an update of an existing variable X, the procedure remains essentially the same as for (4.5), except that the distribution for X' will follow the one for the existing variable X instead of being assigned a $None$ value in the case where all the effects specify empty assignments for the variable:

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

As an example, the output distribution $P(A' \mid r_4 = \{\cdot\}, r_5 = \{A = a_2 \wedge E = e_2\}, A = a_3)$ for the variable A' in Figure 4.3 results in a deterministic distribution with a unique value a_2 with probability 1. If the two rules generate conflicting assignments, the probability mass is divided equally over the alternative values – for instance, the distribution $P(A' \mid r_4{=}\{A{=}a_1\}, r_5{=}\{A{=}a_2 \wedge E{=}e_2\}, A{=}a_3)$ is 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: $P(A' \mid r_4 = \{\cdot\}, r_5 = \{\cdot\}, A = a_3)$ has a unique value a_3 with probability 1.

It is worth noting that such output distribution is entirely parameter-free, as it is directly derived from the specified effects in the rule node. The ordering of the rule parents in the 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 state is detailed in Algorithm 2. The first steps are to extract the input variables for the rule in the Bayesian network (line 1), create a node corresponding to the rule (line 2) and include its conditional dependences (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).

Algorithm 2 : InstantiateProbRule ($\mathcal{B}, r$)

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

1: $\mathcal{I}_r \leftarrow$ input variables for rule r
2: Create chance node R with the rule distribution in Eq. (4.4)
3: Add node R and dependency edges $\mathcal{I}_r \rightarrow R$ to $\mathcal{B}$
4: **if** $Val(R) = \{\cdot\}$ **then**
5: Prune R from $\mathcal{B}$
6: **else**
7: $\mathcal{O}_r \leftarrow$ output variables mentioned in the effects of R
8: **for all** output variable $O \in \mathcal{O}_r$ **do**
9: **if** O' not already in $\mathcal{B}$ **then**
10: Create chance node O' with the output distribution in Eq. (4.5)
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**

4.3.2 Utility rules

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

- 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.

Figure 4.4 illustrates the instantiation of two utility rules r_6 and r_7.

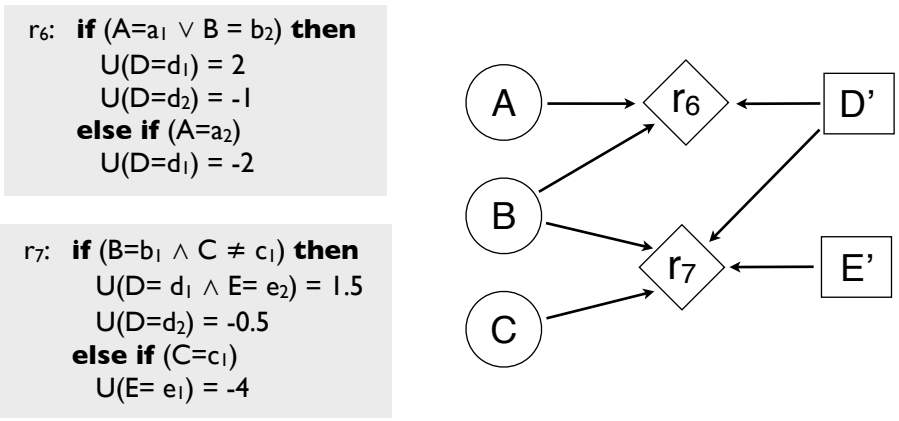


Figure 4.4: Example of instantiation for the two utility rules r_6 and r_7.

The utility distribution associated with each rule is a direct translation of the rule structured if...then...else mapping. Formally, the utility distribution generated by a rule r with input variables $I_1, ...I_n$ and decision variables $A_1, ...A_m$ is defined as:

$$\begin{aligned} &U_r(I_1 = i_1, ...I_n = i_n, A_1 = a_1, ...A_m = a_m) = \\ &\quad U(D_i = \{A_1 = a_1 \wedge ...A_m = a_m\}) \\ &\quad \text{where } i = \min_i(c_i \text{ is satisfied with } I_1 = i_1 \wedge ...I_n = i_n) \end{aligned} \tag{4.7}$$

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

As is conventionally assumed in decision models, the total utility for a given assignment of decision variables is defined as the sum of all utilities. The total utility for the actions $D' = d_1 \wedge E' = e_1$ in the case where $A = a_1$, $B == b_1$ and $C = c_1$ is for instance 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 3 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 at least the utility distribution stipulates a non-zero utility for at least one action (line 4). If not, the node is 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 and utility nodes (line 13).

4.3.3 Application of quantifiers

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

Algorithm 3 : INSTANTIATEUTILRULE ($\mathcal{B}, r$)

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

1: $\mathcal{I}_r \leftarrow$ input variables for rule r
2: Create utility node R with the utility distribution in Eq. (4.7)
3: Add node R and dependency edges $\mathcal{I}_r \rightarrow R$ to $\mathcal{B}$
4: **if** utility distribution is empty for all inputs **then**
5: Prune R from $\mathcal{B}$
6: **else**
7: $\mathcal{D}_r \leftarrow$ decision variables defined for R
8: **for all** decision variable $D \in \mathcal{D}_r$ **do**
9: **if** D' not already in $\mathcal{B}$ **then**
10: Create decision node D' and add it to $\mathcal{B}$
11: **end if**
12: Add in $Val(D')$ all possible values specified for it in the effects of R
13: Add dependency edge $D' \rightarrow R$ to $\mathcal{B}$
14: **end for**
15: **end if**

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). However, the instantiation procedure must be extended in a number of ways to accommodate the presence of underspecified variables. First, the extraction of input variables (line 1 in Algorithms 2 and 3) is not as straightforward, since the random variable names may be underspecified. Second, the internal probability and utility distributions created by the rule must be adapted to account for the presence of quantifiers.

Extraction of input variables

Rules including quantifiers may underspecify the names of random variables. Rule r_3 includes for instance a reference to an underspecified random variable *colour*(o), where o is universally quantified. In order to instantiate the rule, the system must therefore find 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 input variables will therefore be $colour(o_1)$ and $colour(o_2)$.

Algorithm 4 details how this search for matching input variables can proceed. The algorithms starts by extracting the initial input variables for the rule (that may include underspecified variables), and loops on each underspecified variable to find possible groundings in the Bayesian network. The final result is defined as the combination of the fully specified variables for the rule and the possible groundings for the underspecified variables.

Note that use of quantification on the random variable names may lead to a rapid increase in the number of parent variables for the rule node, and must be used with parsimony to ensure the inference problem remains tractable.

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

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

$\mathcal{I}_r \leftarrow$ Initial (possibly underspecified) input variables for rule r
$\mathcal{U}_r \leftarrow$ Subset of variables in $\mathcal{I}_r$ that are underspecified
$\mathcal{G}_r \leftarrow$ possible groundings for underspecified variables $\mathcal{U}_r$, initialised to $[\cdot]$
for underspecified variable $u \in \mathcal{U}_r$ **do**
  for random variable X in $\mathcal{B}$ **do**
    if X matches u **then**
      $\mathcal{G}_r \leftarrow \mathcal{G}_r \cup [X]$
    end if
  end for
end for
return $(\mathcal{I}_r \setminus \mathcal{U}_r) \cup \mathcal{G}_r$

Rule distribution with quantifiers

We have seen in Section 4.3.1 that probability rules were instantiated as a distinct node associated with a rule distribution, and that this node was connected to a set of output variables, each of which defined by an output distribution.

We can extend rule distributions defined in Equation (4.4) to incorporate quantified variables. Let us assume a rule r with input variables $I_1, ...I_n$ and quantified over $\mathbf{x}$. Given a particular assignment for the input variables $I_1, ...I_n$, each rule condition c_i is checked to find all possible groundings of $\mathbf{x}$ that makes c_i satisfied. Note that the number of groundings can be zero if the condition can never be satisfied. The result of this operation is a set of groundings $G = \mathbf{g}_1, ...\mathbf{g}_{|G|}$ where each grounding must satisfy least one condition given the input assignment.[6] Algorithm 5 shows how groundings can be extracted given a rule and a particular input assignment for it. $\phi[a/b]$ denotes (as in mathematical logic) the expression ϕ where all instances of the variable a are substituted by b.

Algorithm 5 : GETGROUNDINGS ($r, \{I_1 = i_1 \wedge ...I_n = i_n\}$)

Input: r: Probability or utility rule
Input: $\{I_1 = i_1 \wedge ...I_n = i_n\}$ assignment for the parent variables of the rule

$G \leftarrow \emptyset$
for $i = 1 \rightarrow n$ **do**
  Let $\mathbf{x}_i$ be the quantified variables in condition c_i
  $G_i \leftarrow \{\text{grounding } \mathbf{g}_k \text{ of } \mathbf{x}_i : \; I_1 = i_1 \wedge ...I_n = i_n \vdash c_i[\mathbf{g}_k/\mathbf{x}_i]\}$
  $G \leftarrow G \cup G_i$
end for
return G

Each grounding $\mathbf{g}_k \in G$ gives rise to a particular effect distribution, and the total effect for the rule is the conjunction of effects $[e_1 \wedge ...e_{|G|}]$ for all possible groundings. The probability of each

[6] This method of handling quantifiers by extracting all possible groundings is an instance of *ground inference* (Getoor and Taskar, 2007). The possible groundings for a given input must be in this setting restricted to a finite set. In order to retain tractability, the maximum number of groundings is also capped by a threshold.

conjunction is defined as the product of probabilities for the grounding-specific effects:

$$P(r=[e_1 \wedge ...e_{|G|}] \mid I_1=i_1, ...I_n=i_n) = \prod_{k=1}^{|G|} P(E_i = e_k[\mathbf{g}_k/\mathbf{x}]) \tag{4.8}$$
$$\text{where } i = \min_i(c_i[\mathbf{x}/\mathbf{g}_k] \text{ is satisfied with } I_1=i_1 \wedge ...I_n=i_n)$$

The effects $e_1 \wedge ...e_{|G|}$ must be fully instantiated (i.e. without any remaining free variables). They can however include conflicting assignments, which are in such case automatically "resolved" in the output distribution by assigning the probability mass uniformly to the alternative values (as explained in the previous section). It should be stressed that the groundings $\mathbf{g}_1, ...\mathbf{g}_{|G|}$ are always extracted *given a specific value assignment* for the input variables. As the number of groundings satisfied for a given input is usually limited to one or two groundings, this instantiation procedure was found to be much more efficient than copying the rule in distinct nodes as initially investigated in Lison (2012b).

Let us analyse an example to show how Equation (4.9) can be applied. Rule r_8 is a rule that quantifies over the variable x:

$$\begin{aligned}
r_8 : \forall x : \\
&\textbf{if } (shape(x) = spherical) \textbf{ then} \\
&\quad \begin{cases} P(graspable(x)=true) = 0.9 \\ P(graspable(x)=false) = 0.1 \end{cases} \\
&\textbf{else if } (shape(x) = conic) \textbf{ then} \\
&\quad \begin{cases} P(graspable(x)=true) = 0.2 \\ P(graspable(x)=false) = 0.8 \end{cases}
\end{aligned}$$

Figure 4.5 shows how the rule r_8 is instantiated in a state with two objects o_1 and o_2, each associated with a random variable describing its shape.

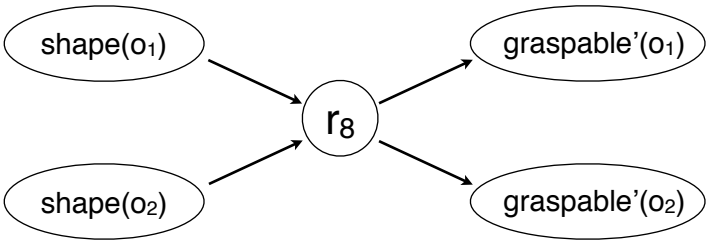


Figure 4.5: Instantiation of the quantified probability rule r_8 on a state with two objects o_1 and o_2.

Given an input assignment such as $shape(o_1) = spherical$ and $shape(o_2) = conic$, the probability distribution $P(r_8 \mid shape(o_1)=spherical, shape(o_1)=conic)$ that reflects the specification in the rule includes a total of four alternative effects:

- $\{graspable(o_1)=true \wedge graspable(o_2)=true\}$ 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 $0.1 \times 0.8 = 0.08$.

We observe that the effects specified for r_8 mention two output variables: $graspable(o_1)$ and $graspable(o_1)$. These two variables are therefore created as children of the node r_8.

Utility distribution with quantifiers

Rule-based utility distributions can be similarly extended to accommodate quantified variables. As for probability rules, quantified utility rules determine a set of groundings that satisfy at least one condition in the rule. Each grounding $\mathbf{g}_k$ generates a particular utility distribution. For a utility rule r with input variables $I_1, ...I_n$, decision variables $A_1, ...A_m$ and quantified variables $\mathbf{x}$, one can find a set of groundings $G = \mathbf{g}_1, ...\mathbf{g}_{|G|}$ that satisfy at least a condition. The total utility distribution is then defined by adding up the utility distributions for each grounding:

$$\begin{aligned} &U_r(I_1 = i_1, ...I_n = i_n, A_1 = a_1, ...A_m = a_m) = \\ &\quad \sum_{k=1}^{|G|} U(D_i = \{A_1 = a_1[\mathbf{g}_k/\mathbf{x}] \wedge ...A_m = a_m[\mathbf{g}_k/\mathbf{x}]\}) \\ &\quad \text{where } i = \min_i(c_i[\mathbf{x}/\mathbf{g}_k] \text{ is satisfied with } I_1 = i_1 \wedge ...I_n = i_n) \end{aligned} \tag{4.9}$$

Rule r_9 illustrates a simple utility rule that includes a quantified variable:

$$\begin{aligned} r_9 : \forall x : \\ &\textbf{if } (graspable(x) = true) \textbf{ then} \\ &\quad \Big\{ U(a_m = grasp(x)) = 2 \\ &\textbf{else} \\ &\quad \Big\{ U(a_m = grasp(x)) = -1 \end{aligned}$$

The result of instantiating rule r_9 on a state with two objects o_1 with associated random variables $graspable(o_1)$ and $graspable(o_2)$ is shown in Figure 4.6.

For a given input such as $graspable(o_1) = true \wedge graspable(o_2) = false$, two groundings $x = o_1$ and $x = o_2$ are found. The total utility distribution is then straightforwardly derived to be:

- $U_r(graspable(o_1) = true, graspable(o_2) = false, a_m = grasp(o_1)) = 2$
- $U_r(graspable(o_1) = true, graspable(o_2) = false, a_m = grasp(o_2)) = -1$

4.4 Processing workflow

The two previous sections have detailed how probability and utility rules are internally defined, and how they can be instantiated as nodes of a dynamic decision network. We are now ready to

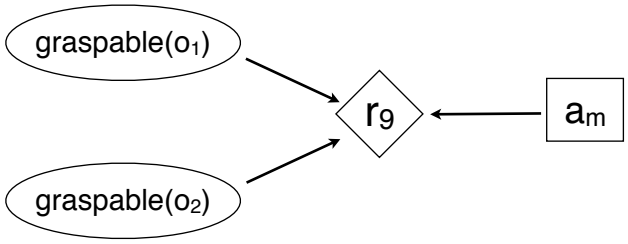


Figure 4.6: Instantiation of the quantified utility rule r_9 on a state with two objects o_1 and o_2.

demonstrate how collections of rules can be applied 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.

We start by describing how the various rules describing a dialogue domain can be organised in collections called "models". We then show how these models are triggered to update state variables and express the utility of particular actions. We also demonstrate the general processing workflow on a detailed example.

4.4.1 Domain specification

To ease the specification of dialogue domains, the probabilistic rules are grouped into collections called *models*. A model is simply a collection of rules that is associated with one or more "trigger" variables. The update of such trigger variables in the dialogue state leads to the instantiation of all the rules included in the model. Formally, a model m is defined as a pair $\langle \mathcal{T}_m, \mathcal{R}_m \rangle$ where $\mathcal{T}_m$ is a set of variables names that trigger the instantiation of the model, and $\mathcal{R}_m$ are the rules included in the model.

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 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. Many reasoning tasks can be structured in terms of probabilistic rules. Beyond the usual probability and utility models for dialogue management, probabilistic rules have also been applied to tasks related to natural language understanding and generation. As an example, the dialogue domain specified for the experiments described in Chapter 7 include 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 (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 each other in a principled framework grounded in probabilistic inference. 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).

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.

Joint optimisation: Finally, the use of a unified modelling formalism allows the multiple domain models to be optimised jointly instead of tuning each model in isolation. 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).

4.4.2 Update algorithm

The software architecture adopted in this thesis takes the form of an event-driven, blackboard architecture 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 general procedure to update the dialogue state upon the reception of new variables is provided in Algorithm 6.

State update is performed whenever a chance occurs on one or more state variables, due to e.g. to new inputs processed by the ASR or NLU component. The first step of the algorithm is to incorporate the new variable in the dialogue state and possibly relate them to their predicted values (lines 2-3 of Algorithm 6, see below for more explanation). The algorithm the triggers the instantiation of the relevant domain models (line 4), leading to a chain of updates. If the expanded dialogue state contains decision and utility variables, the algorithms then searches for the optimal action, selects it, and activates the models that are triggered as a result of this action (lines 6-8). Finally, outdated and intermediary nodes are pruned away from the expanded state and the evidence is incorporated in the state distribution (line 10).

We now describe each of these steps in detail.

Algorithm 6 : UPDATESTATE ($\mathcal{B}, \mathbf{X}$)

Input: $\mathcal{B}$: Bayesian network for the current state
Input: $\mathbf{X}$: New random variables 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($\mathcal{B}, \mathbf{e}$)

Connection between predictions and observations

Some models are 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 in the dialogue state 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 domains often include observations that are themselves uncertain and must therefore rely on some type of soft evidence mechanism (Pan et al., 2006). The method adopted in this thesis is to add a new chance node, subsequently called the *equivalence node* eq_X, with boolean values that is conditionally dependent on both X and X_p.[8]. The conditional probability distribution of eq_X is deterministic:

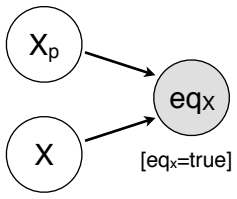


Figure 4.7: Equivalence node eq_X with parents X and X_p.

$$P(eq_X = true \mid X = x, X_p = x_p) = \begin{cases} 1 & \text{if } x = x_p \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

The assignment $eq_X = true$ is added to the evidence. The posterior distribution given the evidence can then be straightforwardly derived to allow the prediction to act as a prior for the

[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, and effectively prime the user actions that are most likely given a particular conversational situation.

[8] The addition of a distinct node to express the evidence is necessary in our case since both X and X_p can have arbitrary incoming and outgoing edges with other variables in the dialogue state.

observed distribution.

$$
\begin{aligned}
& P(X = x \,|eq_X = true) \\
& = \alpha \; P(X = x) \sum_{x_p \in Val(X_p)} P(eq_X = true | X = x, X_p = x_p) P(X_p = x_p) && (4.11) \\
& = \alpha \; P(X = x) \; P(X_p = x) && (4.12)
\end{aligned}
$$

Algorithm 7 enumerates the steps involved in the integration of predicted values upon the update of the set of random variables $Vars$.

Algorithm 7 : IntegratePredictions ($\mathcal{B}, \mathbf{e}, Vars$)

1: **for all** $Var \in Vars$ **do**
2: **if** there is a corresponding prediction variable $Var_p \in \mathcal{B}$ **then**
3: Create equivalence node eq_{Var} with distribution in Eq. (4.10)
4: Insert eq_{Var} in $\mathcal{B}$ with parents Var and Var_p
5: Add assignment $[eq_{Var} = true]$ 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 (if applicable), the next step is to trigger the relevant domain models . Algorithm 8 summarises the steps involved in the instantiation of the domain models following an update of the dialogue state. 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. Once all models are checked, the output variables of the instantiated rules become updated variables, and the procedure is repeated until no more model can be applied. To avoid infinite triggering cycles, a model can only be instantiated once per update.

Action selection

If the dialogue state expanded with the instantiated rules contain utility and decision nodes, the system must in this case decide on the action to perform. Algorithm 9 illustrates how actions can be automatically selected on the basis of the current dialogue state augmented with decision and utility variables. The algorithm simply searches for the assignment of action values that maximise the current utility given the dialogue state and the evidence, and returns it. The utility nodes are removed from the state once the decision is made. Note that the action selection procedure shown in Algorithm 9 only takes into account the current (immediate) utility and does not rely on any forward planning. Chapter 6 will demonstrate how this action selection mechanism can be expanded to perform online planning on a limited horizon.

Algorithm 8 : 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 $\mathcal{B}, \mathbf{e}$

Algorithm 9 : 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 in the dialogue state results in the integration of multiple new nodes in the dialogue state. This monotonic expansion of the dialogue state may quickly lead to tractability problems if left unchecked. The last step of state update is therefore to reduce the dialogue state to its minimal size, by removing all intermediary nodes – such as 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 attached to the equivalence nodes is also integrated in the posterior distribution of the state variables.

The procedure is outlined in Algorithm 10. 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 in the set of nodes to keep.

Figure 4.8 illustrates the input and output of the pruning process. Only the nodes A', B, C, D and E' are retained. Note that that the primes are deleted from the random variable names.

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

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

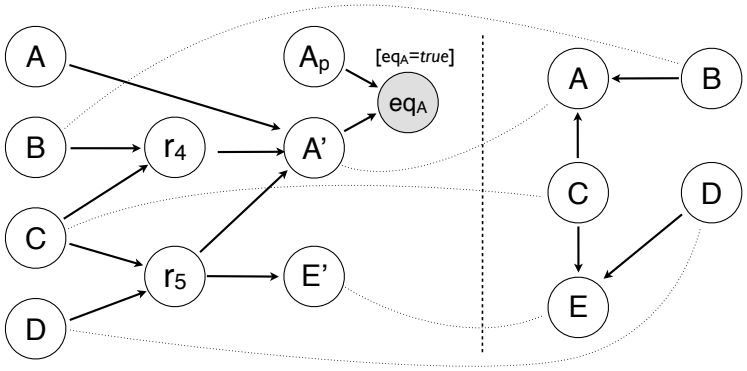


Figure 4.8: Illustration of the state pruning process. The left side shows a dialogue state before pruning, and the right side the corresponding state following the pruning operation. The dotted lines denote the correspondence between nodes.

4.4.3 Detailed example

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 certain of 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: Model m_1 is triggered by a_u and includes two utility rules r_{10} and r_{11}:

$$
\begin{aligned}
r_{10}: \; & \forall x: \\
& \textbf{if } (a_u = Request(x)) \textbf{ then} \\
& \Big\{ U(a_m = Move(x)) = 2 \\
& \textbf{else} \\
& \Big\{ U(a_m = Move(x)) = -2 \\
\\
r_{11}: \; & \Big\{ U(a_m = AskRepeat) = 0.5
\end{aligned}
$$

Rule r_{10} 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_{11} assign a utility of 0.5 for asking a clarification question.

Model m_2: Model m_2 is triggered by a_m and includes the predictive rule r_{12}:

$$
\begin{aligned}
r_{12}: \; & \forall x: \\
& \textbf{if } (a_m = AskRepeat \wedge a_u = x) \textbf{ then} \\
& \Big\{ P(a_{u-p} = x) = 0.9
\end{aligned}
$$

Rule r_{12} specifies that the probability that the user will repeat his last utterance when asked by the system to do so amounts to 0.9.

Processing workflow

Figure 4.9 details the steps involved in the state update after receiving dialogue act hypotheses from the natural language understanding component. The example covers the following short interaction:

USER : Now move forward
$\tilde{a}_u$: $\langle$(*MoveForward*, 0.6), (*MoveBackward*), 0.4)$\rangle$
SYSTEM : Could you please repeat?
USER : Please move forward!
$\tilde{a}_u$: $\langle$(*MoveForward*, 0.7), (*MoveLeft*, 0.2), (*MoveBackward*, 0.1) $\rangle$
SYSTEM : OK, moving forward!

Step 1 inserts the new dialogue act hypotheses on the dialogue state. This insertion triggers model m_1 since the model contains a_u as trigger variable. The model 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

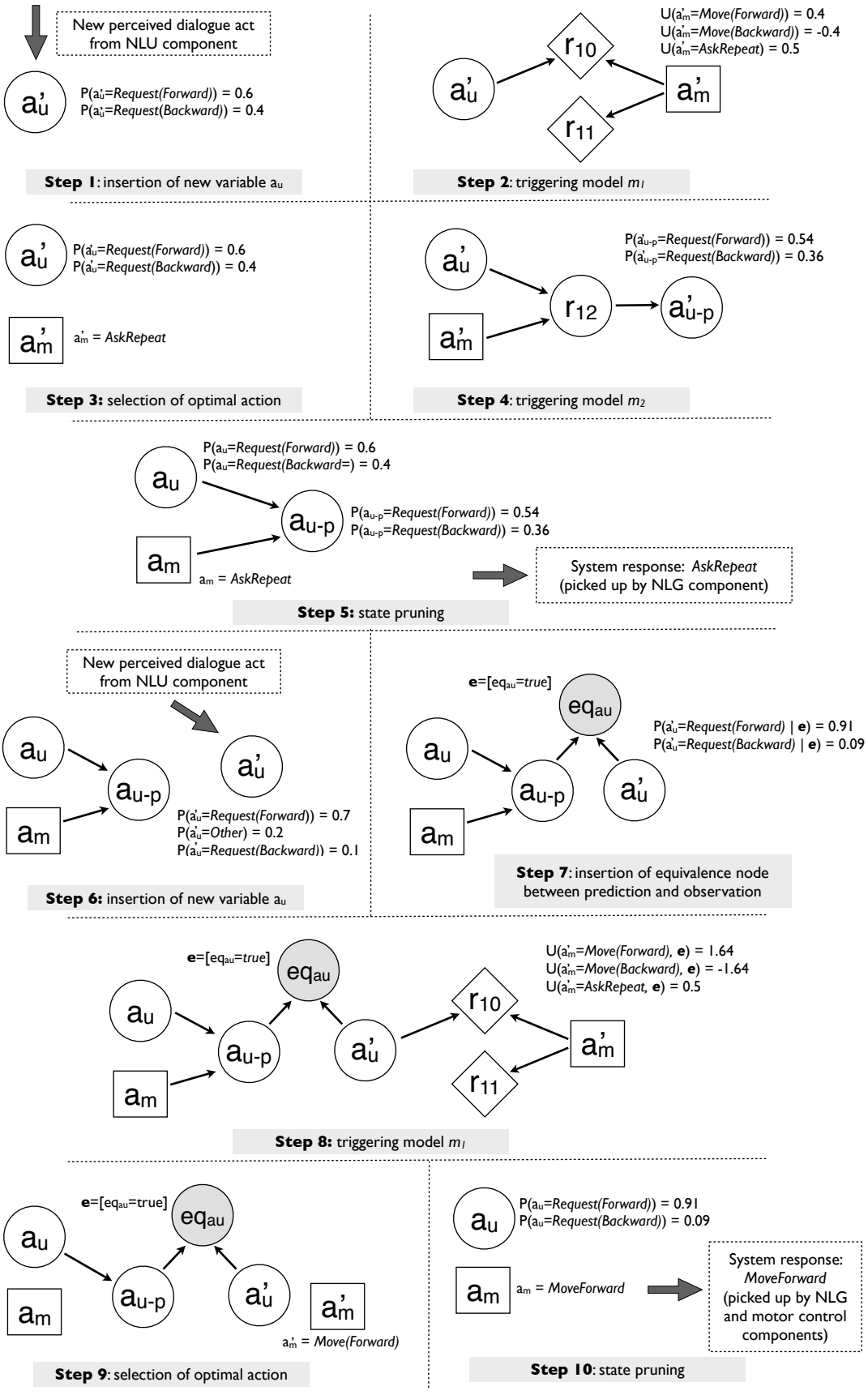


Figure 4.9: Detailed example of processing workflow.

model m_2 in Step 4, which creates a prediction node a'_{u-p} expressing the expected values 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. The model m_2 is triggered upon this selection but rule r_{12} only generates an empty effect and is therefore directly deleted. Finally, the state is pruned of its intermediary nodes in Step 10, leaving only the last user and system actions a_u and a_m.

4.5 Advanced modelling

Dialogue domains often need to represent random variables whose values are expressed via specific data structures such as collections or strings. The rule-based formalism described in the previous sections must be extended to efficiently operate on these data structures. We first describe how conditions and effects can be defined on variables that represent collections, and then discuss how rules can manipulate strings.

4.5.1 Operations on collections

Some state variables are best represented as collections of elements. For instance, the dialogue state may include variables that enumerate e.g. 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 the set of possible elements.

Both the conditions and effects of probabilistic rules can be adapted to include special-purpose operations for collections:

- Rule conditions can include checks on the presence or absence of particular elements in a collection, such as $a \in A$ or $a \notin A$.
- Rule effects can also be augmented to manipulate elements from a collection. 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 collection), *remove effects* (deleting an element from a collection) and *clear effects* (clearing all elements of a collection). The resulting collections are sorted by insertion order.

Figure 4.10 illustrates two rules that make use of add and remove effects to update the distribution defined for the state variable A.

Output distributions $P(X' = x' \mid r_1 = e_1, ...r_n = e_n)$ must be adapted to take into account the three types of effects. Let $\mathbf{e}$ denote as before the conjunction of all effects $e_1 \wedge ...e_n$. In addition to the classical set of values $\mathbf{e}(X)$ assigned for the variable X, the distribution also refers to the set of values $\mathbf{e}_{add}(X)$ and $\mathbf{e}_{remove}(X)$ that are to be respectively added and removed from the variable

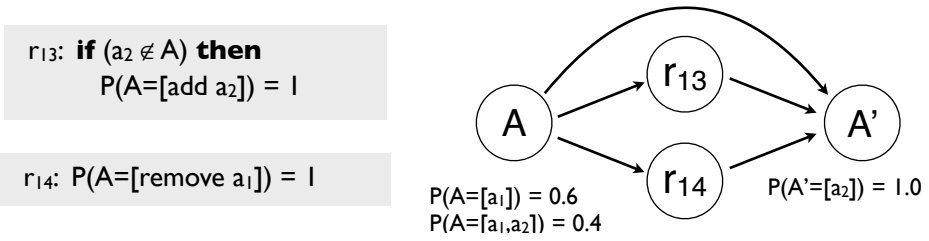


Figure 4.10: Example of rules using add/remove effects to manipulate collections.

X. A boolean $\mathbf{e}_{clear}(X)$ indicates whether at least an effect requires the existing values for X to be cleared.

Given these definitions, the output distribution defined in Equation (4.5) for a new output variable can then be rewritten as:

$$P(X'=x' \mid r_1=e_1, ...r_n=e_n) = \begin{cases} \frac{\sum_{v \in \mathbf{e}(X)} \mathbf{1}(x'=v)}{|\mathbf{e}(X)|} & \text{if } \mathbf{e}(X) \neq \emptyset \\ \mathbf{1}(x' = \mathbf{e}_{add}(X)) & \text{if } \mathbf{e}(X)=\emptyset \text{ and } \mathbf{e}_{add}(X) \neq \emptyset \\ \mathbf{1}(x' = None) & \text{otherwise} \end{cases} \tag{4.13}$$

The traditional assignment effects in $\mathbf{e}(X)$ and the add effects $\mathbf{e}_{add}(X)$ are mutually incompatible (in case both $\mathbf{e}(X)$ and $\mathbf{e}_{add}(X)$ are non-empty, the assignment effects are by convention assumed to take precedence).

In case the output variable replaces an existing variable, the output distribution becomes:

$$P(X'=x' \mid r_1=e_1, ...r_n=e_n, X=x) = \begin{cases} \frac{\sum_{v \in \mathbf{e}(X)} \mathbf{1}(x'=v)}{|\mathbf{e}(X)|} & \text{if } \mathbf{e}(X) \neq \emptyset \\ \mathbf{1}(x' = [\mathbf{e}_{add}(X) \cup [x \setminus \mathbf{e}_{remove}(X)]]) & \text{if } \mathbf{e}(X)=\emptyset \text{ and } \neg\, \mathbf{e}_{clear}(X) \\ \mathbf{1}(x' = \mathbf{e}_{add}(X)) & \text{if } \mathbf{e}(X)=\emptyset, \mathbf{e}_{add}(X) \neq \emptyset \text{ and } \mathbf{e}_{clear}(X) \\ \mathbf{1}(x' = None) & \text{otherwise} \end{cases} \tag{4.14}$$

4.5.2 Operations on strings

Many of the data structures present in the dialogue state are strings – including for instance the user utterance u_u and the system utterance u_m. It is therefore useful to provide special-purpose functionalities for manipulating strings within the conditions and effects of probabilistic rules. 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. The template can include slots to fill, and the values extracted for these slots can be locally reused in the effects of the rule.

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

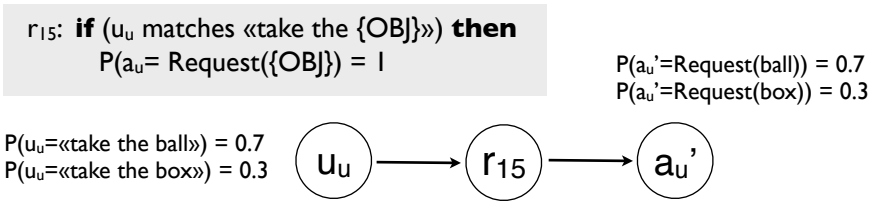


Figure 4.11: Example of rule using string matching operations.

Such string matching functionality can be accommodated by adapting the rule distribution in Equation (4.4) in the following manner:

$$P(r=e \mid I_1=i_1, \ldots I_n=i_n) = P(E_i = e[\mathbf{s}_i \backslash \mathbf{x_i}]) \tag{4.15}$$
$$\text{where } i = \min_i(c_i \text{ is satisfied with } I_1=i_1 \wedge \ldots I_n=i_n) \text{ with filled slots } \mathbf{s}_i$$

The careful reader many notice that the above equation is similar to the rule distribution for quantified rules, with the exception that only one "grounding" (corresponding here to the filled slots) is extracted for a given input assignment.

4.6 Related work

Heriberto's relational state: Cuayáhuitl (2011) an hierarchical aproach (Cuayáhuitl, 2011).

Otterlo2012

parameter sharing

ref to reinforcement learning based on same idea of if then else lists

link with PDDL, PPDDL, RPDDL!

incrementality!

4.7 Conclusion

way to structure a dynamic decision network, hence a POMDP

Chapter 5

Learning from Wizard-of-Oz data

where do the numbers come from?

5.1 Bayesian parameter estimation

5.1.1 Key idea

5.1.2 Parameter priors

talk about simplifying assumptions: we are learning from partial data

5.1.3 Approximate inference

5.2 Estimation of action utilities from Wizard-of-Oz data

Sell it as some type of imitation learning?

5.2.1 Data representation

5.2.2 Integrating the evidence

5.3 Experiments

5.3.1 Wizard-of-Oz data collection

5.3.2 Experimental setup

5.3.3 Empirical results

5.3.4 Analysis

5.4 Conclusion

Chapter 6

Learning from interactions

6.1 Bayesian Reinforcement Learning

[1]

6.1.1 Model-free methods

6.1.2 Model-based methods

6.2 Online planning

6.3 Experiments

6.3.1 Wizard-of-Oz data collection

6.3.2 User simulator

6.3.3 Experimental setup

6.3.4 Empirical results

6.3.5 Analysis

6.4 Conclusion

[1] Interestingly, offline and online approaches to planning are not mutually exclusive, but can be combined together to offer "the best of both worlds". The idea is to perform offline planning to precompute a rough policy, and use this policy as a heuristic approximation to guide the search of an online planner (Ross et al., 2008). These heuristic approximations can for instance be used to provide lower and upper bounds on the value function, which can be exploited to prune the lookahead tree.

Chapter 7

User evaluation

ANOVA?

Chapter 8

Concluding remarks

8.1 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).

8.2 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

Appendix A

Relevant probability distributions

Uniform distribution

Multinomial distribution

Normal distribution

Dirichlet distribution

Kernel distribution

Should we include the last one?

Appendix B

Domain specification for user trials

put here a summary of the probabilistic rules applied in the last experiment (user evaluation)

Appendix C

The openDial toolkit

Similarity to Olympus, Jaspis, Ariadne dialogue architectures?

Bibliography

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. ISBN 978-1-60558-168-2.

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.

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 John H. L. Hansen and Bryan 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. ISBN 0387310738.

D. Bohus and A. I. Rudnicky. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23(3):332–361, July 2009.

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. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.4547.

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. ISBN 978-3-540-70880-3. URL http://dl.acm.org/citation.cfm?id=1760700.1760716.

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. ISBN 978-1-4673-4651-1.

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. doi: 10.1007/978-3-642-30353-1_24. URL http://dx.doi.org/10.1007/978-3-642-30353-1_24.

J. Choi and K.-E. Kim. Inverse reinforcement learning in partially observable environments. *Journal of Maching Learning Research*, 999999:691–730, 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. Lossless value directed compression of complex user goal states for statistical spoken dialogue systems. In ISCA (2011), pages 1029–1032.

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. Learning dialogue agents with bayesian relational state representations. In *Proceedings of the IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems (IJCAI-KRPDS)*, pages 9–15. IJCAI, 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, 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, 2012.

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 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. Morgan Kaufmann, 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, 2008.

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.

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.

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.

L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar. Probabilistic relational models. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

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 Gregory F. Cooper and Serafín Moral, editors, *UAI*, pages 211–219. Morgan Kaufmann, 1998. ISBN 1-55860-555-X.

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.

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.

ISCA, editor. *Proceedings of the 12th Annual Conference of the International Speech Communication Association (Interspeech 2011)*, 2011.

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. ISBN 0470060263, 9780470060261.

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.

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.

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.

S. Kopp, B. Jung, N. Lessmann, and I. Wachsmuth. Max - a multimodal assistant in virtual reality construction. *Künstliche Intelligenz*, 17(4):11, 2003.

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.

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.

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. ISSN 1351-3249.

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. ISSN 1351-3249. doi: 10.1017/S1351324900002539. URL http://portal.acm.org/citation.cfm?id=973935.973943.

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. ISSN 1063-6676.

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, 2010.

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 dialogue management in relational domains. In *SLTC Workshop on Action, Perception and Language (APL)*, Lund, Sweden, October 2012b.

P. Lison. Probabilistic dialogue models with prior domain knowledge. In *Proceedings of the SIGDIAL 2012 Conference*, pages 179–188, 2012c.

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. (accepted).

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 Publishers Inc., 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.

M. F. McTear. *Spoken dialogue technology - toward the conversational user interface*. Springer, 2004.

J.-J. Ch. Meyer and W. Van Der Hoek. *Epistemic logic for AI and computer science*, volume 41. Cambridge University Press, 2004.

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.

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. Morgan Kaufmann Publishers Inc. ISBN 1-55860-614-9.

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.

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.

R. Pan, Y. Peng, and Z. Ding. Belief update in bayesian networks using uncertain evidence. In *18th IEEE International Conference on Tools with Artificial Intelligence*, pages 441–444, 2006.

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.

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, Carnegie Mellon University, Pittsburgh, PA, USA, 2004. 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.

ShaoWei 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.

M. Poesio and H. Rieser. An incremental model of anaphora and reference resolution based on resource situations. *Dialogue & Discourse*, 2(1):235–277, 2011.

P. Poupart. *Exploiting structure to efficiently solve large scale partially observable markov decision processes*. PhD thesis, University of Toronto, Toronto, Canada, 2005.

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, 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.

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, 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.

Stephane Ross, Joelle Pineau, Sebastien Paquet, and Brahim Chaib-Draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, July 2008.

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.

S. J. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Pearson Education, 2010.

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.

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. ISBN 978-1-4244-1746-9.

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. Morgan Kaufmann Publishers Inc.

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. ISSN 1387-2532.

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, 2010.

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. ISBN 0-262-51112-6.

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.

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, 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.

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.

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. 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 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. doi: 10.1016/j.csl.2006.06.008.

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.

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. Morgan Kaufmann Publishers Inc.

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.