Available online at www.sciencedirect.com

ScienceDirect

Computer Speech and Language 34 (2015) 232–255

COMPUTER SPEECH AND LANGUAGE

www.elsevier.com/locate/csl

# A hybrid approach to dialogue management based on probabilistic rules☆

Pierre Lison *

*Language Technology Group, Department of Informatics, University of Oslo, P.O. Box 1080 Blindern, N-0316 Oslo, Norway*

Received 23 May 2014; received in revised form 1 October 2014; accepted 12 January 2015
Available online 21 January 2015

**Abstract**

We present a new modelling framework for dialogue management based on the concept of *probabilistic rules*. Probabilistic rules are defined as structured mappings between logical conditions and probabilistic effects. They function as high-level templates for probabilistic graphical models and may include unknown parameters whose values are estimated from data using Bayesian inference. Thanks to their use of logical abstractions, probabilistic rules are able to encode the probability and utility models employed in dialogue management in a compact and human-readable form. As a consequence, they can reduce the amount of dialogue data required for parameter estimation and allow system designers to directly incorporate their expert domain knowledge into the dialogue models.

Empirical results of a user evaluation in a human–robot interaction task with 37 participants show that a dialogue manager structured with probabilistic rules outperforms both purely hand-crafted and purely statistical methods on a range of subjective and objective quality metrics. The framework is implemented in a software toolkit called *OpenDial*, which can be used to develop various types of dialogue systems based on probabilistic rules.


*Keywords:* Spoken dialogue systems; Dialogue management; Probabilistic graphical models; Bayesian inference; Human–robot interaction

## 1. Introduction

The design of dialogue strategies is a challenging task in the development of spoken dialogue systems (SDS). The selection of system actions is often grounded in a complex dialogue state encompassing a variety of factors such as the dialogue history, the user goals and preferences, the external context and the task to perform. In addition, spoken dialogue is also riddled with uncertainties arising from speech recognition errors, ambiguous inputs, partially observable environments, and unpredictable dialogue dynamics. These difficulties are particularly striking in the case of human–robot interaction (HRI). By their very definition, human–robot interactions take place in a physical, situated environment that must be captured and monitored by the robotic agent. They must also typically deal with high levels

☆ This paper has been recommended for acceptance by R.K. Moore.

* Tel.: +47 96799812.
*E-mail address:* plison@ifi.uio.no

http://dx.doi.org/10.1016/j.csl.2015.01.001
0885-2308/

of noise and uncertainty caused by e.g. imperfect sensors and actuators. The robot's tracking of the current dialogue state is therefore bound to remain partial and error-prone.

Two families of dialogue management approaches have been historically developed to address these issues. The first family relies on hand-crafted strategies, ranging from finite-state automata to more complex inference procedures based on formal logic and classical planning. These strategies provide principled techniques for the interpretation and generation of dialogue moves on the basis of the dialogue participants' mental states (including their shared knowledge). Dialogue is then framed as a collaborative activity in which the interlocutors work together to coordinate their actions, maintain a shared conversational context, resolve open issues and satisfy social obligations (Allen et al., 2000; Larsson, 2002; Jokinen, 2009). Such approaches can yield detailed analyses of various dialogue behaviours, but they generally assume complete observability of the dialogue state and provide only a limited account of errors and uncertainties. In addition, the knowledge bases from which the system's decisions are derived must be completely specified in advance by domain experts. Their deployment in practical applications is thus non-trivial.

The second family relies on statistical modelling techniques (Levin et al., 2000; Roy et al., 2000; Young et al., 2010; Rieser and Lemon, 2011). The dialogue is here represented as a stochastic control process – often a *Markov decision process* (MDP) or a *Partially observable Markov decision process* (POMDP) – and the optimal dialogue strategy is the one that maximises the system's long-term expected utility. These probabilistic models offer an explicit account for the various uncertainties that can arise during the interaction. They also allow the dialogue strategies to be optimised in a data-driven manner instead of relying on hand-crafted mechanisms, making it easier to adapt to new environments or users. However, these probabilistic models typically depend on large amounts of training data to estimate their parameters – a requirement that is hard to satisfy for most dialogue domains. This shortage of relevant datasets is especially critical in human–robot interactions, given the high costs of collecting and annotating dialogue data for these dialogue domains.

This article presents a hybrid approach to dialogue management that seeks to combine the benefits of hand-crafted and statistical techniques in a single framework. As in previous work on POMDPs models for dialogue management, the approach represents the dialogue state as a Bayesian network that is regularly updated with new observations and employed to derive the system's actions. However, the domain models are no longer expressed with traditional factored representations but are instead structured via *probabilistic rules*. As explained in the next pages, the rules can be viewed as high-level templates for probabilistic graphical models. The use of probabilistic rules provides an efficient *abstraction layer* that allows the system designer to capture the domain models in a concise and human-readable form.

The present article is structured as follows. Section 2 reviews the key principles of dialogue management, focusing in particular on MDP- and POMDP-based approaches. Section 3 outlines the formalism of probabilistic rules and their instantiation as nodes of a graphical model. Section 4 describes how the parameters of probabilistic rules can be estimated via Bayesian inference, using either Wizard-of-Oz data (supervised learning) or user interactions (reinforcement learning). Section 5 presents the *OpenDial* toolkit, a domain-independent dialogue toolkit that allows dialogue developers to construct dialogue systems using probabilistic rules. Section 6 describes a user evaluation of this modelling approach in a human–robot interaction domain. Section 7 contrasts the framework with related work. Finally, Section 8 concludes the article and reviews future research directions.

## 2. Dialogue management

### 2.1. System architecture

The general architecture of a spoken dialogue system is depicted in Fig. 1. The user speech signals are first processed by the speech recogniser, resulting in a list of recognition hypotheses $\tilde{u}_u$, where each hypothesis is associated to a particular probability or confidence score.[1] Dialogue understanding then maps these hypotheses into high-level semantic representations of the dialogue act expressed by the user. These dialogue acts are also expressed as a list $\tilde{a}_u$ of semantic hypotheses together with their respective probabilities. Dialogue management is then in charge of selecting the best system action to perform given the current conversational context. The dialogue manager outputs a particular action $a_m$

[1] Throughout this article, we follow the convention of denoting user-specific variables with the subscript $u$ and machine-specific variables with the subscript $m$.

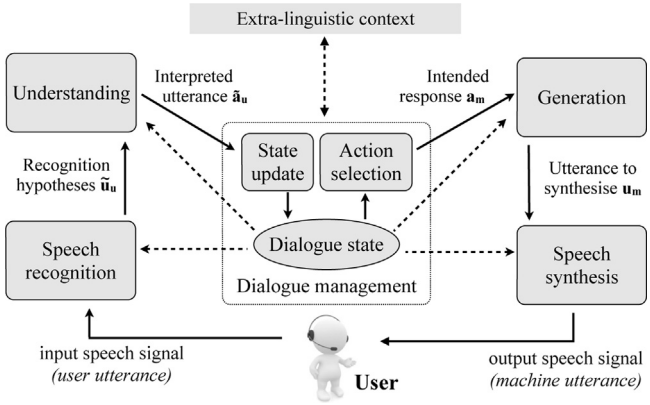



Fig. 1. Information flow in spoken dialogue systems. The solid lines between system components denote the main input and outputs while the dashed lines represent optional contextual information.

(which can be void, i.e. leading to no action). If the selected action relates to a communicative act, language generation is triggered to find its best linguistic realisation, denoted $u_m$. Finally, the constructed system utterance is sent to a speech synthesiser in order to generate the corresponding audio signal.

Dialogue management serves a double function within this processing pipeline, illustrated by the two boxes in Fig. 1. The first function is to maintain a representation of the current *dialogue state*. The dialogue state is a representation of what is known of the current conversational situation (from the system's point of view) and is often factored in several variables related to the dialogue history, the external context, and the tasks to perform. This dialogue state is regularly updated on the basis of new observations (in the form of e.g. new user utterances or changes in the external context). The second function of dialogue management is to decide which actions to undertake in a particular dialogue state. Dialogue management is therefore essentially a problem of *decision-making under uncertainty*: the system is provided with multiple observations (which may be partial or erroneous) about the current state of the interaction and must find the best action to execute in this state. We describe in the next section how such decision problems can be formalised in statistical terms.

### 2.2. Statistical approaches to dialogue management

#### 2.2.1. Markov decision processes

Statistical approaches to dialogue management typically represent interactions as a Markov decision process (MDP) $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$ where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $T$ the transition function that encodes the probability $P(s' | s, a)$ of reaching state $s'$ after executing action $a$ in state $s$, and $R$ the reward function that expresses the reward value associated with the execution of action $a$ in state $s$. The state space corresponds to the set of possible dialogue states, while the transition function captures the internal dynamics of the conversation, indicating how the dialogue state is expected to change as a result of the system actions. Finally, the reward function $R$ represents the particular objectives and costs of the application domain.

Given a particular MDP, the goal is to find a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maps each 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, which corresponds to the expected long-term accumulation of rewards from the current state up to a given horizon. As the transition model is usually unknown, various methods have been devised to automatically extract this optimal policy from experience via reinforcement learning. Due to the high number of cycles necessary to converge onto a near-optimal policy, interactions with real users are often impractical. Instead, most 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 dialogue data extracted from recordings of human–human dialogues or Wizard-of-Oz experiments (Pietquin, 2008; Frampton and Lemon, 2009; Rieser and Lemon, 2011). The user simulator allows the agent to explore millions of dialogue trajectories on a scale that would be impossible to achieve with real users. Simulated interactions, however, run the risk of deviating from actual user

behaviours. They are also more difficult to apply in human–robot interaction (and other types of situated dialogue domains) due to the need to capture the physical environment in addition to the user.

#### 2.2.2. Partially observable Markov decision processes

A limitation faced by MDP approaches is the assumption that the dialogue state is fully observable, making it difficult to account for uncertainties arising from e.g speech recognition errors. A solution is to extend the MDP framework by viewing the state as a hidden variable that is indirectly inferred from observations. Such an 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 between states, and $R$ the reward function. However, the actual state is no longer directly observable. Instead, the process is associated with an observation space $\mathcal{O}$ that expresses the set of possible observations that can be perceived by the system (for instance, the N-best lists of user dialogue acts). The function $Z$ defines the probability $P(o|s)$ of observing $o$ in state $s$.

In the POMDP setting, the system's knowledge at a given time is represented by the *belief state*, which is a probability distribution $P(s)$ over all possible states and is often factored as a Bayesian network (Bui et al., 2009; Thomson and Young, 2010). The belief state is continuously updated as new observations become available. Given a dialogue system with current belief state $b$ that executes a system action $a$ followed by observation $o$, the updated belief $b'$ is directly derived from Bayes' rule:

$$b'(s) = P(s' \mid a,\ o) = \eta P(o \mid s') \sum_{s \in \mathcal{S}} P(s' \mid s,\ a) b(s) \tag{1}$$

where $\eta$ is a normalisation factor. The observation $o$ can for instance correspond to the N-best list $\tilde{a_u}$ of dialogue act hypotheses. A POMDP policy is then defined as a function mapping each possible belief state to its optimal action. As for MDP-based methods, POMDP approaches often derive the dialogue policy from interactions with a user simulator (Young et al., 2010; Daubigney et al., 2012), although some recent approaches also explored the use of direct interactions (Gašić et al., 2011, 2013). The optimisation process required to extract POMDP policies is, however, considerably more complex than for MDPs, as the belief state space is continuous and high-dimensional. Approximation techniques are therefore necessary in order to extract dialogue policies of reasonable quality, such as grid-based discretisations (Young et al., 2010), linear function approximation (Thomson and Young, 2010; Daubigney et al., 2012) or non-parametric methods based on Gaussian processes (Gašić et al., 2013).

## 3. Probabilistic rules

One major bottleneck in statistical approaches to dialogue management is the size of the parameter space. The framework presented in this article seeks to reduce the numbers of parameters by taking advantage of expert knowledge about the dialogue domain. More precisely, the framework rests on the idea of representing the transition and utility models of a dialogue POMDP in terms of *probabilistic rules*. These rules are practically defined as *if...then...else* constructions that map logical conditions to probabilistic effects, based on the following skeleton:

$\forall$ **x**,
**if** (condition 1 holds) **then**
  Distribution 1 over possible effects
**else if** (condition 2 holds) **then**
  Distribution 2 over possible effects
...
**else**
  Distribution $n$ over possible effects

Each *if...then* branch specifies both a condition (expressed as a logical formula) and an associated distribution over possible effects. The *if...then...else* construction is read in sequential order, as in programming languages, until a satisfied condition is found, which causes the activation of the corresponding probabilistic effects. The conditions

and effects of the rule may include underspecified variables, denoted **x**, which are universally quantified on top of the rule.[2] The mapping between conditions and effects specified by the rule is in this case duplicated for every possible assignment (grounding) of the underspecified variables, allowing the system designer to abstract over particular aspects of the domain and express the dialogue models with a small number of rules.

In line with previous work on POMDP approaches to dialogue management, the dialogue state is represented as a Bayesian network composed of state variables representing various aspects of the current context. The probabilistic rules are then applied at runtime to update this dialogue state on the basis of new observations and select the system actions. As we will see, this process is performed by instantiating the rules as latent nodes in the graphical model.

We first present how such rules can be used to express probability distributions, and then show how to generalise the formalism to utility functions. We shall use the notion of *probabilistic rules* as an umbrella term that covers all types of rules, while *probability rules* will only refer to rules expressing probability distributions, and *utility rules* to rules expressing utility functions.

### 3.1. Probability rules

#### 3.1.1. General structure

A probability rule is formally expressed as an ordered list of branches $[br_1, \ldots, br_n]$ in which $br_i$ denotes the $i$-th branch of the *if...then...else*. Each branch $br_i$ is a pair $\langle c_i, P(E_i)\rangle$ where $c_i$ is a logical condition and $P(E_i)$ is a categorical probability distribution over a set of mutually exclusive effects. The random variable $E_i$ represents the effect of the rule at branch $i$ and has a set of possible values $Val(E_i) = \{e_{i,1}, \ldots, e_{i,m_i}\}$, where $m_i$ is the number of effects defined for the branch. Each effect $e_{i,j}$ has a corresponding probability denoted $\theta_{i,j}$. These probabilities must satisfy the standard probability axioms.[3]

Given these elements, a probability rule reads as such:

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

#### 3.1.2. Conditions

The conditions $c_1, \ldots, c_n$ are expressed as logical formulae over a subset of variables included in the Bayesian network representing the dialogue state. The logical formulae can be constructed using the usual operators from predicate logic (conjunctions, disjunctions, and negations) as well as various binary relations (equality, inequalities, set membership, etc.). The terms of the logical formulae may also include underspecified (i.e. free) variables which

[2] The variables **x** are variables in the sense of first-order logic and should not be confused with the random variables of the probabilistic model.

[3] In other words, the probability values must satisfy to $0 \leq \theta_{i,j} \leq 1$ for all values $i, j$ and $\sum_{j=1}^{m_i} \theta_{i,j} = 1$ for every branch of a given rule.

are universally quantified on the top of the rule.[4] The variables employed in the conditions define the *input variables* of the rule. An example of a rule condition is $(a_u = x \wedge a_m = AskRepeat)$, which is satisfied when the variable $a_u$ equals an underspecified value $x$ and the variable $a_m$ equals the value *AskRepeat*. This particular condition contains two input variables, $a_u$ and $a_m$. Rule conditions can be arbitrarily complex and include nested formulae.

The conditions offer a compact partitioning of the state space. Without such a partitioning, a rule ranging over the input variables $I_1, \ldots, I_k$ would need to enumerate $Val(I_1) \times \ldots \times Val(I_k)$ possible conditions. The reliance on the *if. . .then. . .else* structure reduces this number to $n$ partitions, where $n$ corresponds to the number of branches of the rule and is usually small. Partitioning is closely related to the notions of state abstraction and state aggregation in planning and reinforcement learning (Li et al., 2006). State abstraction/aggregation corresponds to mapping a large state space into a more compact, abstract space (corresponding here to the rule conditions).

### 3.1.3. Effects

Associated to each condition $c_i$ is a collection of mutually exclusive effects $e_{i,1}, \ldots, e_{i,m_i}$. Each effect $e_{i,j}$ represents a specific assignment of values for a set of variables called the *output variables* of the rule. We adopt the convention of denoting output variables with a prime to distinguish them from the input variables. An example of an effect is $\{a'_u = x\}$, which is an assignment of the variable $a_u$ to the value $x$. Effects can be void (that is, represent no assignment) and can also express assignments covering more than one output variable. When no terminating **else** branch is explicitly specified at the end of a rule, a final **else** associated with a void effect is implicitly assumed to ensure that the partitioning is exhaustive.

Each effect $e_{i,j}$ is assigned a probability $\theta_{i,j} = P(E_i = e_{i,j})$. The probabilities can be either fixed by hand by the system designer or estimated empirically (as explained in Section 4). For convenience, if the sum of all effect probabilities is lower than 1, we assume that the remaining probability mass is assigned to a void effect.

As already noted, the probabilistic rules allow specific elements **x** inside the conditions and effects of a rule to be *underspecified*. Based on this quantification mechanism, the rules can cover large portions of the state space in a compact manner. One advantage of this representation is that it allows for powerful forms of *parameter sharing*, since the effect probabilities $\theta_{i,j}$ are made independent of the various instantiations of the variables **x** (see below for an example).

### 3.1.4. Examples

$$\forall x, \quad \textbf{if}\,(a_u = x \wedge a_m = AskRepeat)\,\textbf{then} \quad \{P(a'_u = x) = 0.9 \tag{$r_1$}$$

Rule $r_1$ offers a prediction on the next user dialogue act $a'_u$ based on the last dialogue act $a_u$ and system's action $a$. has two input variables: $a_u$ and $a_m$ and one output variable: $a'_u$. The rule stipulates that if the system asks the user to repeat the last utterance $x$, the user is expected to comply and repeat $x$ with probability 0.9. A void effect is implicitly associated with the remaining probability mass (0.1 in this example). The universal quantification allows us to express that this probability is independent of the particular dialogue act $x$ uttered by the user.

$$\begin{aligned} \forall x, \quad & \textbf{if}\,(a_u = RequestAction(x) \wedge a_m = Do(x))\,\textbf{then} \\ & \{P(a'_u = Confirm) = 0.2 \\ & \textbf{else if}\,(a_u \neq RequestAction(x) \wedge a_m = Do(x))\,\textbf{then} \\ & \begin{cases} P(a'_u = Disconfirm) = 0.5 \\ P(a'_u = RequestAction(Stop)) = 0.3 \end{cases} \end{aligned} \tag{$r_2$}$$

The rule $r_2$ distinguishes between two cases. If the user requests a particular action $x$ to be performed and the system executes it, the user is expected to utter a positive confirmation feedback with probability 0.2. Else, if the system

[4] Logical quantifiers are only allowed at the top level of a rule, and cannot therefore appear in the inner part of a rule condition or effect. This limitation is introduced to ensure that conditions can be checked efficiently, under real-time constraints. Similar restrictions can be found in planning formalisms (McDermott et al., 1998).

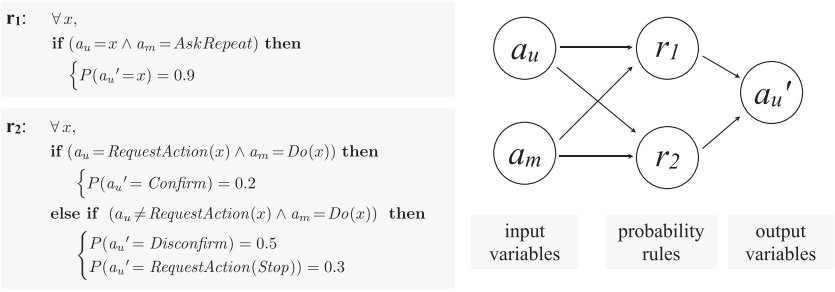



Fig. 2. Graphical model resulting from the instantiation of the two probability rules $r_1$ and $r_2$.

performs an action $x$ that was uncalled for, the user is expected to utter a disconfirmation feedback with probability 0.5 or ask the system to stop with probability 0.3.

### *3.1.5. Instantiation*

Probability rules can be used to provide a compact encoding for the transition model of the given dialogue domain. As in previous POMDP approaches to dialogue management, this transition model is applied to update the Bayesian network representing the current (belief) dialogue state upon the reception of new observations or the execution of system actions. To this end, the probability rules are converted at runtime into standard probability distributions. This is done by instantiating each rule as a latent node that connects the rule's input variables (i.e. the variables that are mentioned in the rule conditions) to the rule's output variables (i.e. the variables that are mentioned in the rule effects). This latent node represents a random variable on the possible effects of the rule. Fig. 2 illustrates this instantiation on a dialogue state comprising two variables: $a_u$ (last dialogue act from the user) and $a_m$ (last dialogue act from the system). The two rules $r_1$ and $r_2$ are applied onto this dialogue state, leading to the creation of two rule nodes and a common output variable $a'_u$ (predicted user dialogue act for the next time step).

The conditional probability distribution of a rule node $r$ given its input variables $I_1, \ldots, I_k$ is formally defined as:

$$\begin{aligned} &P(r = e \mid I_1 = i_1, \ldots, I_k = i_k) = P(E_i = e) \\ &\quad \text{where } i = \min(\{i : c_i \text{ is satisfied with } I_1 = i_1 \wedge \ldots \wedge I_k = i_k\}) \end{aligned} \tag{3}$$

A condition $c_i$ is said to be satisfied iff the conditional assignment logically entails that the condition is true, that is: $(I_1 = i_1 \wedge \ldots \wedge I_k = i_k) \vdash c_i$. If the rule contains universally quantified variables, the possible groundings are first extracted on the basis of the rule inputs and instantiated inside the conditions and effects of the rule. The rule conditions are checked in sequential order until one condition is satisfied. Since the last condition $c_n$ corresponds to the final **else** branch and is therefore trivially true, there will always be at least one satisfied condition.

For example, the conditional probability distribution $P(r_2 | a_u = SayHi, a_m = Do(TurnLeft))$ is a categorical distribution with three values: $\{a'_u = Disconfirm\}$ with probability 0.5, $\{a'_u = RequestAction(Stop)\}$ with probability 0.3, and the void effect $\{\cdot\}$ with probability 0.2.

Output variables (such as $a'_u$ in the example) are conditionally dependent on all rule nodes that refer to them in their effects. Their conditional distribution is a direct reflection of the combination of effects specified in the rule nodes that are parents of the output variable. Let $X'$ denote an arbitrary output variable with parent rules $r_1, \ldots r_n$. Given a particular assignment of effects for these rules, the conditional distribution is expressed as:

$$P(X' = x \mid r_1 = e_1, \ldots, r_n = e_n) = \begin{cases} \dfrac{\text{Number of assignments } \{X' = x\} \text{ in } e_1, \ldots, e_n}{\text{Number of assignments for } X' \text{ in } e_1, \ldots, e_n} & \text{if at least one effect } e_i \text{ is not void} \\ \mathbf{1}(x = None) & \text{otherwise} \end{cases} \tag{4}$$

where $\mathbf{1}$ is the indicator function. In other words, the distribution for the variable $X'$ follows from the effects of the parent nodes. The distribution $P(a'_u \mid r_1 = \{a'_u = SayHi\}, r_2 = \{\cdot\})$ in Fig. 2 has for instance a single value *SayHi* with

probability 1. If the effects include conflicting assignments, the distribution is spread uniformly over the alternative values. If all effects are void, the output variable is set to a default *None*.

As shown in Fig. 2, the main difference between traditional transition models and probability rules is the introduction of an abstraction layer (the rule nodes) that mediates between the input variables (state at time $t$) and output variables (state at time $t+1$). The introduction of this abstraction layer allows the system designer to decompose a complex transition model into smaller parts, each part being captured by a distinct probability rule.

### 3.2. Utility rules

In addition to tracking the current dialogue state, the dialogue manager must also perform action selection, i.e. find the action with the highest expected utility in the current state. The formalism outlined in the previous section can also be used to express utility functions with only minor notational changes.

#### 3.2.1. General structure

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

$$
\begin{array}{l}
\forall \mathbf{x}, \\
\quad \textbf{if } (c_1) \textbf{ then} \\
\quad\quad \left\{ \begin{array}{l} U_1(d_{1,1}) = \theta_{1,1} \\ \ldots \\ U_1(d_{1,m_1}) = \theta_{1,m_1} \end{array} \right. \\
\quad \textbf{else if } (c_2) \textbf{ then} \\
\quad\quad \left\{ \begin{array}{l} U_2(d_{2,1}) = \theta_{2,1} \\ \ldots \\ U_2(d_{2,m_2}) = \theta_{2,m_2} \end{array} \right. \\
\quad \ldots \\
\quad \textbf{else} \\
\quad\quad \left\{ \begin{array}{l} U_n(d_{n,1}) = \theta_{n,1} \\ \ldots \\ U_n(d_{n,m_n}) = \theta_{n,m_n} \end{array} \right.
\end{array}
\tag{5}
$$

A utility rule associates utility values to system decisions. As for probability rules, the conditions $c_i$ are defined as logical formulae over a set of input variables. The decisions $d_{i,j}$ are assignments of specific values to decision variables, and the corresponding utilities $\theta_{i,j}$ are arbitrary real numbers – which, contrary to probabilities, may be positive or negative.

#### 3.2.2. Examples

$$
\begin{array}{ll}
\forall x, & \textbf{if } (a_u = \mathit{RequestAction}(x)) \textbf{ then} \\
& \quad \left\{ U(a'_m = \mathit{Do}(x)) = 5 \right. \\
& \textbf{else} \\
& \quad \left\{ U(a'_m = \mathit{Do}(x)) = -5 \right.
\end{array}
\tag{$r_3$}
$$

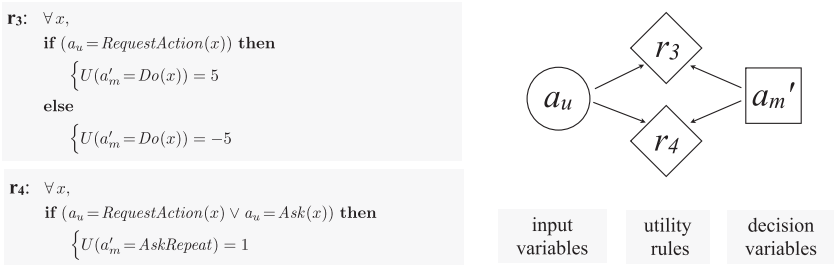



Fig. 3. Graphical model resulting from the instantiation of the two utility rules $r_3$ and $r_4$. Decision nodes are depicted by squares and utility nodes by diamonds.

Rule $r_3$ states that the utility of the system action $a'_m = Do(x)$ will be 5 if the action $x$ is indeed requested for by the user, and $-5$ if the action was not requested.

$$\forall x, \quad \textbf{if}\,(a_u = RequestAction(x) \vee a_u = Ask(x))\,\textbf{then} \quad \{U(a'_m = AskRepeat) = 1 \tag{r4}$$

Rule $r_4$ states the utility of the clarification request $a'_m = AskRepeat$ is set to 1, provided the last user utterance is a request or a question. The underspecified variable $x$ enables us to specify that this utility is independent of the particular user request or question.

### 3.2.3. Instantiation

Utility rules are instantiated in the dialogue state according to a procedure similar to probability rules. Each utility rule is translated into a utility node that is dependent both on the input and decision variables for the rule. Fig. 3 illustrates this process.

The utility function associated with each rule can be straightforwardly derived from the definition of the rule. Formally, the utility function generated by a rule $r$ with input variables $I_1, \ldots, I_k$ and decision variable $D'$ is defined as:

$$\begin{aligned} &U_r(D' = d \mid I_1 = i_1, \ldots, I_k = i_k) = U_i(D' = d) \\ &\quad \text{where } i = \min(\{i : c_i \text{ is satisfied with } I_1 = i_1 \wedge \ldots \wedge I_k = i_k\}) \end{aligned} \tag{6}$$

In other words, the utility of the action $D' = d$ is equal to its value in the utility table $U_i$, where $i$ corresponds to the first branch in the rule whose condition is satisfied. If no utility is explicitly specified for $D' = d$ in the utility table, the default value is zero. As is conventionally assumed in graphical models, the total utility for a particular action is defined as the *sum* of utilities resulting from each utility node. As an illustration, if we assume that the variable $a_u$ of Fig. 3 is a categorical distribution with two values *RequestAction*(*Stop*) with probability 0.7 and *Other* with probability 0.3, the expected utility of action $a'_m = Do(Stop)$ will be equal to $5 \times 0.7 - 5 \times 0.3 = 2$.

## 4. Parameter estimation

The examples of probabilistic rules shown in the previous section relied on probabilities and utilities fixed by hand. In practical dialogue domains, such parameters are often difficult to determine manually, due to the inherent unpredictability that characterise spoken dialogue. They are therefore best estimated empirically from actual dialogue data. We developed two alternative methods for parameter estimation: a supervised learning approach based on Wizard-of-Oz data, and a reinforcement learning approach from real or simulated interactions. Both methods assume that the structure of probabilistic rules (the mapping between conditions and effects) is provided by the system designer, while the rule parameters (the effect probabilities and utilities) are determined via statistical estimation. This "division of labour" between the human designer and the learning algorithm is motivated by the fact that system designers often have a good grasp of the domain structure and relations between variables but are typically unable to quantify the precise probability of an effect or utility of an action.

Bayesian inference is employed in both cases to estimate the best values for the rule parameters on the basis of observed dialogue data. The key principle in Bayesian learning is to associate each parameter with a prior distribution

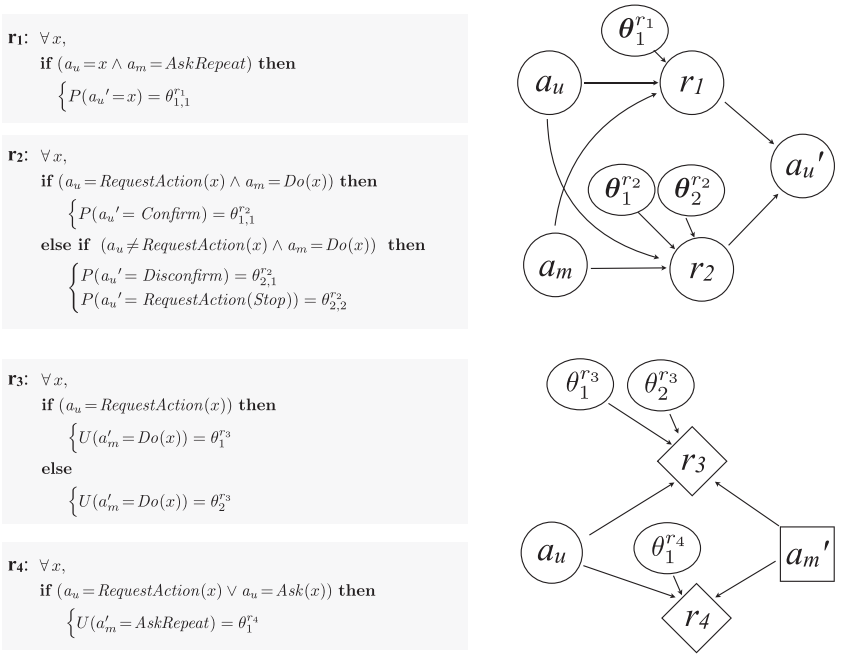



Fig. 4. Examples of instantiation of probabilistic rules with unknown parameters. The parameter variables $\boldsymbol{\theta}_1^{r_1}$, $\boldsymbol{\theta}_1^{r_2}$ and $\boldsymbol{\theta}_2^{r_2}$ (corresponding to probability values), are multivariate continuous distributions typically encoded by Dirichlet priors, while the parameter variables $\theta_1^{r_3}$, $\theta_2^{r_3}$ and $\theta_1^{r_4}$ (corresponding to utilities) are univariate continuous distributions encoded by normal or Gaussian priors.

over its range of possible values, and gradually refine this distribution as data points are processed. For probability rules, we have seen in Eq. (2) that each branch $i$ of a given rule is associated with a set of effect probabilities $\theta_i = \{\theta_{i,1}, \ldots, \theta_{i,m_i}\}$, where $m_i$ corresponds to the number of effects specified in the branch. If these probabilities $\boldsymbol{\theta}_i$ are unknown, we can define a prior distribution $P(\boldsymbol{\theta}_i)$ over their (continuous) range of values. Similarly, for utility rules, each unknown utility $\theta_i$ is associated to a prior distribution $P(\theta_i)$. These random variables are instantiated in the graphical model as distinct nodes connected to their corresponding rules via outgoing edges. This process is illustrated in Fig. 4.

Various types of priors can be used to encode these parameter distributions. For probability rules, unknown probabilities can be represented by Dirichlet priors. Since Dirichlet distributions are the conjugate priors of multinomial and categorical distributions (Koller and Friedman, 2009), they constitute a convenient choice to express the parameters of probability rules (recall from Section 3.1 that the rule effects are represented as categorical distributions). Formally, a Dirichlet distribution is a continuous, multivariate probability distribution encoded with a set of hyper-parameters $\alpha_1, \ldots \alpha_k$, where $k$ is the dimensionality of the Dirichlet. For each rule branch $i$ containing $m_i$ alternative effects with unknown probabilities, a Dirichlet distribution of dimensionality $m_i$ will be constructed. The $\alpha$ counts for the Dirichlet

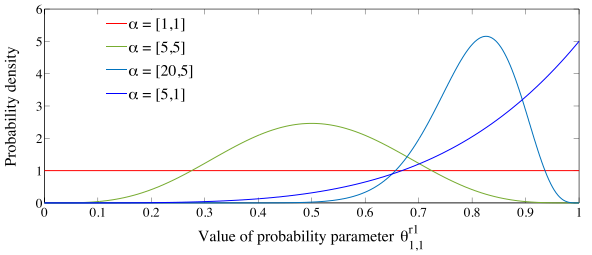



Fig. 5. Examples of Dirichlet priors for the unknown parameter $\boldsymbol{\theta}_1^{r_1}$ specified in rule $r_1$ under various values of the $[\alpha_1, \alpha_2]$ hyper-parameters. In this two-dimensional case, the Dirichlet distribution is equivalent to a Beta distribution. The second dimension corresponds to the probability of the implicit void effect in the rule.

prior can be used to reflect the designer's initial assumptions regarding the relative frequency of each effect. Fig. 5 shows a range of Dirichlet priors for the probability value $\theta^{r_1}_{1,1}$ used in the rule $r_1$.

For utility rules, uniform or Gaussian distributions are similarly applied to express the priors over utility parameters. Each utility is associated with its own prior distribution (contrary to probabilities, utilities are independent from one another and do not need to sum up to 1).

On the basis of these parameter priors, the goal of the estimation process is to calculate their posterior distributions given the available data. This calculation is performed through standard techniques for (exact or approximate) probabilistic inference on directed graphical models. We describe below two distinct parameter estimation methods, respectively based on supervised and reinforcement learning.

### 4.1. Estimation via supervised learning

The most straightforward parameter estimation method relies on supervised learning from examples of expert behaviour collected via Wizard-of-Oz experiments. Wizard-of-Oz interactions are interactions between human users and a dialogue system which is remotely controlled by a human expert operating "behind the curtains". They constitute a simple and efficient method to collect realistic conversational behaviour in the absence of a fully implemented or optimised system. We represent Wizard-of-Oz interactions as a sequence of state–action pairs $\mathcal{D} = \{\langle \mathcal{B}_i, a_i \rangle : 1 \leq i \leq n\}$, where $\mathcal{B}_i$ corresponds to the dialogue state (encoded as a Bayesian network) at time $i$, and $a_i$ is the associated action performed by the wizard.

Given a collection of Wizard-of-Oz examples $\mathcal{D}$ and a set of probabilistic rules with unknown parameters $\boldsymbol{\theta}$, the learning goal is to estimate the posterior distribution $P(\boldsymbol{\theta} \mid \mathcal{D})$ over the rule parameters given the observed data. Using Bayes' rule and assuming the examples in $\mathcal{D}$ are independent and identically distributed (i.i.d.), this posterior distribution can be decomposed as:

$$P(\boldsymbol{\theta} \mid \mathcal{D}) = \eta \, P(\boldsymbol{\theta}) \prod_{\langle \mathcal{B}_i, a_i \rangle \in \mathcal{D}} P(a_i \mid \mathcal{B}_i ; \boldsymbol{\theta}) \tag{7}$$

where $P(a_i \mid \mathcal{B}_i ; \boldsymbol{\theta})$ represents the likelihood of the wizard selecting the action $a_i$ in the dialogue state $\mathcal{B}_i$ under a particular assignment of values for the rule parameters $\boldsymbol{\theta}$. In order to define this likelihood distribution, we shall assume that the wizard is a (mostly) rational agent and will tend to select actions that are deemed most useful in their respective state. In other words, the utility $U(a_i \mid \mathcal{B}_i ; \boldsymbol{\theta})$ is expected to be ranked highly relative to the utility of other possible actions. Formally, we express the likelihood of observing the wizard executing action $a_i$ in a dialogue state $\mathcal{B}_i$ given the parameters $\boldsymbol{\theta}$ as a geometric distribution:

$$\begin{aligned} P(a_i \mid \mathcal{B}_i ; \boldsymbol{\theta}) &= \begin{cases} \eta \, p & \text{if } a_i \text{ is the action with highest utility in } U(a \mid \mathcal{B}_i ; \boldsymbol{\theta}) \\ \eta(1-p)p & \text{if } a_i \text{is the action with second-highest utility in } U(a \mid \mathcal{B}_i ; \boldsymbol{\theta}) \\ \dots \end{cases} \\ &= \eta(1-p)^{x-1} \, p \text{ where } x \text{ is the rank of } a_i \text{ in } U(a \mid \mathcal{B}_i ; \boldsymbol{\theta}) \end{aligned} \tag{8}$$

The $\eta$ factor in the above distribution is a normalisation factor, while the probability $p$ represents the learner confidence in the rationality of the wizard. The probability $p$ indirectly defines the learning rate of the estimation process: the higher the probability, the faster the learner will converge to a policy that imitates the wizard action (but at the cost of a higher vulnerability to the occasional errors and inconsistencies on the part of the wizard).

In practice, the Bayesian learning algorithm operates by traversing the state–action pairs one by one, in a single pass, and updating the posterior parameter distributions after each pair: $P(\boldsymbol{\theta}_{(i+1)}) = \eta \, P(\boldsymbol{\theta}_{(i)}) \, P(a_i \mid \mathcal{B}_i ; \boldsymbol{\theta}_{(i)})$. As more Wizard-of-Oz examples are processed, the parameter distributions gradually narrow down their spread and converge to the values that best imitate the conversational behaviour of the wizard. Given the complexity of the resulting probabilistic models (which include both discrete and continuous distributions as well as partially observed data), sampling techniques such as likelihood weighting are employed to approximate the posteriors (Fung and Chang, 1989). It is worth noting that in the above approach, the utility of a given action $a$ represent the total, long-term utility for the agent of executing $a$ in a particular state. It corresponds therefore to the notion of $Q$-value in the reinforcement learning literature (or equivalently, to a reward value with a planning horizon limited to the present step). The idea of directly estimating

utility values from examples of expert behaviour can also be found in imitation or demonstration-based learning [see e.g. Billard et al., 2008].

### 4.2. Estimation via reinforcement learning

Supervised learning techniques are hampered by the necessity to collect Wizard-of-Oz data for the domain. An alternative to supervised learning is to let the dialogue system learn the best conversational behaviour via trial and error from its own interaction experience – that is, through reinforcement learning (RL) – without relying on the provision of external examples. In Lison (2013), we outlined a Bayesian reinforcement learning approach to the estimation of rule parameters. As in the supervised learning case, the internal models of the domain (i.e. the transition and reward models) are expressed through probabilistic rules and associated with a number of parameters. In the course of repeated interactions with a real or simulated user, the agent gradually refines the spread of these parameter distributions. These parameters are then subsequently employed at runtime to plan the optimal action to perform, taking into account every possible source of uncertainty (including state uncertainty, stochastic action effects, and uncertainty over the parameter values). Experiments with a user simulator demonstrated that a transition model structured by probabilistic rules was able to converge to a high-quality dialogue policy (measured in terms of average return per episode) much faster than a traditional factored model. The interested reader is invited to consult Lison (2013, 2014) for more details.

## 5. Implementation

The framework presented in this article is fully implemented in an open-source software toolkit called *OpenDial*.[5] The toolkit, which is released under an MIT license, is a Java-based, domain-independent platform for the development of spoken dialogue systems based on the specification of probabilistic rules in XML format.

Listing 1 illustrates an example of a probability rule (corresponding to rule $r_2$ in Section 3) encoded in XML. Unknown parameters (probabilities or utilities) can be replaced by identifiers in the rules for the purpose of parameter estimation.

```
<rule>
    <case>
        <condition>
                <if var="a_u" value=RequestAction({X})"/>
                <if var="a_m" value=Do({X})"/>
        </condition>
        <effect prob="0.2">
                <set var="a_u" value="Confirm"/>
        </effect>
    </case>
    <case>
        <condition>
                <if var="a_u" relation="!=" value="RequestAction({X})"/>
                <if var="a_m" value="Do({X})"/>
        </condition>
        <effect prob="0.5">
                <set var="a_u" value="Disconfirm"/>
        </effect>
        <effect prob="0.3">
                <set var="a_u" value="RequestAction(Stop)"/>
        </effect>
    </case>
</rule>
```

Fig. 5. Listing 1. Example of a probability rule in the *OpenDial* XML format. Terms surrounded by curly brackets (such as {X} in this example) denote underspecified variables.

[5] The toolkit and its documentation are available at http://opendial-toolkit.net.

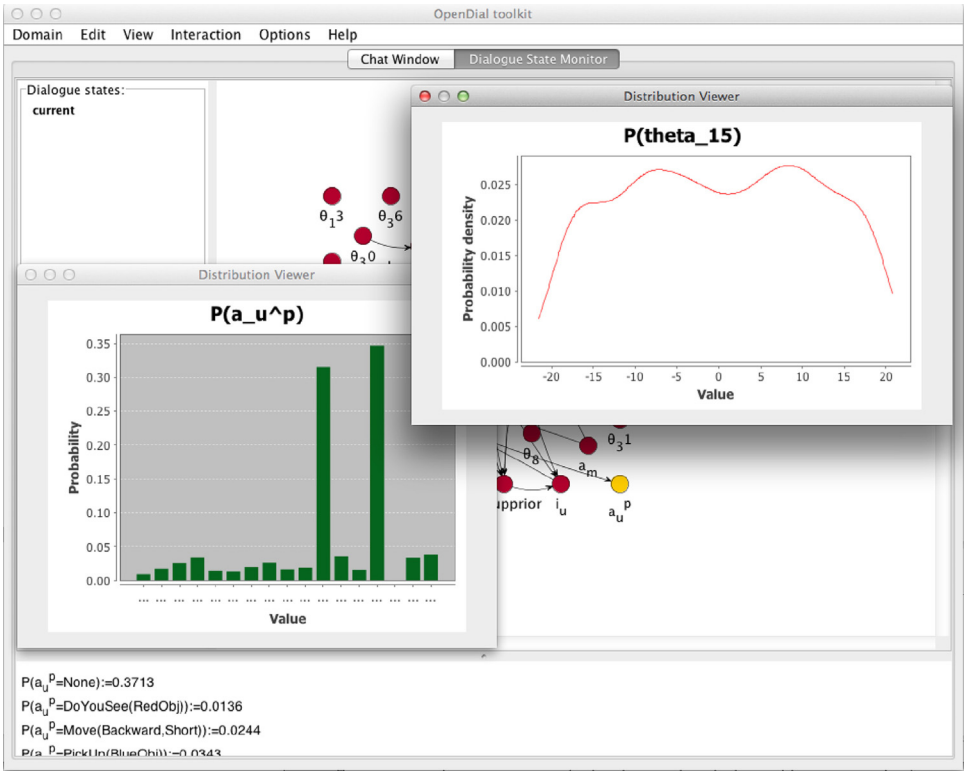



Fig. 6. Graphical interface for the *OpenDial* toolkit.

As in information-state approaches to dialogue management (Larsson and Traum, 2000; Bos et al., 2003), *OpenDial* relies on a blackboard architecture, with the dialogue state serving as the central information hub for the system. Probabilistic rules and external modules are connected to this state and read/write to it as they process their information flow. The flexible nature of blackboard architectures allows components to be plugged in and out of the system without affecting the rest of the pipeline (Corkill, 1989). In addition to the data structures and algorithms for probabilistic inference, parameter estimation and planning, *OpenDial* also comes bundled with a collections of plug-ins that extend the toolkit with additional modules for e.g. speech recognition and synthesis. Finally, a graphical interface (Fig. 6) allows the system designer to interactively evaluate dialogue strategies and inspect the dialogue state at any time.

## 6. User evaluation

A user experiment was conducted to evaluate the practical performance of probabilistic rules compared to existing modelling approaches. The aim of the experiment was to compare three alternative approaches to dialogue management in a human–robot interaction domain. The three approaches respectively correspond to:

- a purely hand-crafted approach, based on a finite-state automaton
- a purely statistical approach, based on factored statistical models
- a hybrid approach based on probabilistic rules

The evaluation metrics include both objective metrics extracted from the interaction logs and subjective metrics of user satisfaction derived from a survey completed by the participants after each interaction.

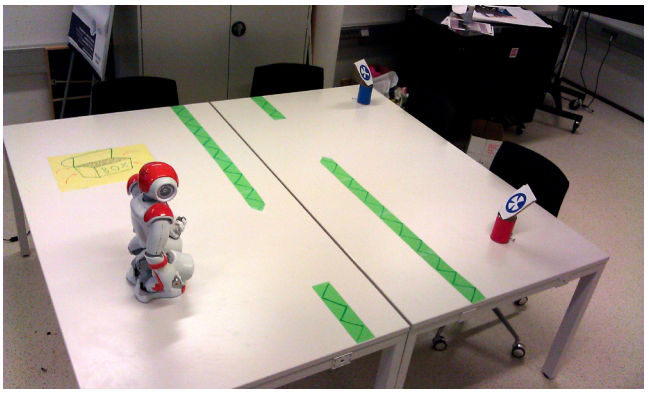


Fig. 7. Interaction scenario for the experiment. Two graspable objects (one red and one blue) stand on the right side of the table. Imaginary walls (in green) are also placed on the table, as well as a yellow landmark (on the left side of the table) depicting the final destination of the robot.

### *6.1. Interaction scenario*

The interactions revolved around a human user and a Nao robot[6] placed on a table along with two graspable objects, as shown in Fig. 7. The objective of the interactions was to instruct the robot to perform the following sequence of tasks:

1 Walk to the other end of the table without bumping into the imaginary walls,
2 Pick up the designated object,
3 Bring the object back to the left side of the table,
4 And finally release it on the yellow landmark.

The robot was able to perceive the presence, colour and location of each physical object thanks to visual markings placed at their top. It could also grasp the objects with the help of permanent magnets fixed inside the robot's hands. An off-the-shelf speech recogniser (Nuance Vocon 3200) was used to recognise the spoken commands of the participants. The speech recogniser was coupled to four microphones integrated in the robot's head and used a small hand-crafted grammar (with 21 context-free rules) as language model. As often the case in human–robot interaction, speech recognition errors occurred frequently, notably due to the noise from the robot's motors and the large distance between the human user and the microphones. The language understanding module consisted of a simple pattern matching mechanism and mapped utterances to logical representations of the user's dialogue acts.

A total of 18 distinct instructions can be executed by the robot in this interaction scenario, such as walking in different directions, picking up and releasing objects, or enumerating the objects currently perceived by the robot's vision modules. The dialogue understanding module could recognise a total of 25 distinct user dialogue acts $a_u$ (including the aforementioned instructions and a number of additional dialogue acts for grounding and engagement), while the system actions $a_m$ had a total of 41 possible values, which comprised both physical movements and a range of clarification requests. The dialogue state $\mathcal{B}$ designed for the dialogue domain is factored into nine distinct state variables, encoding both the user intentions and actions, the external context (e.g. objects perceived by the robot), the recent interaction history, and the physical state of the robot (e.g. whether the robot is currently in motion). The total size of the joint state space amounts to $335 \times 10^6$ distinct states.

A small corpus of Wizard-of-Oz interactions was collected prior to the user trials in order to estimate the parameters of the dialogue management models (see next section). We recorded a total of 10 interactions, each with a distinct participant. All interactions were performed in English. The participants to the Wizard-of-Oz study (5 males and 5 females) were selected amongst students and employees from the University of Oslo. The interactions were encoded as a sequence of pairs $\langle \mathcal{B}_i, a_i \rangle$, where each system turn $i$ is represented by the selected wizard action $a_i$ and the dialogue state $\mathcal{B}_i$ in effect at the time the selection was made. The interactions had on average 84.2 system turns. 39% of these

[6] The Nao robot is a humanoid robot produced by Aldebaran Robotics (http://www.aldebaran-robotics.com).

turns resulted in a void action (i.e. no action at all selected by the wizard), 40% in a physical action, and the remaining 21% in a verbal response such as a factual answer or a clarification request.

### 6.2. Dialogue management approaches

To evaluate how the choice of a modelling framework affects the interaction quality, we developed three alternative dialogue managers for the domain: a purely hand-crafted dialogue manager based on a finite-state automaton, a purely statistical strategy based on factored distributions, and a hybrid strategy using probabilistic rules.

*Approach 1: Hand-crafted model*

The first dialogue manager uses a finite-state automaton to determine the current conversational situation and its corresponding system action. The automaton contains 16 distinct states and is triggered upon each new user dialogue act $a_u$. Attached to each edge is a logical condition that determines when the edge can be traversed. The conditions are defined on the basis of the last two user dialogue acts complemented by two contextual variables respectively describing the objects currently perceived or carried by the robot.

To account for noisy inputs, the finite-state automaton relied on three threshold values that divide the confidence of the user dialogue acts into four regions (respectively corresponding to unreliable, weak confidence, moderate confidence, and high confidence results). These threshold values were determined empirically on the basis of the actual probability values generated by the speech recogniser and NLU module during the Wizard-of-Oz study.

*Approach 2: Factored statistical model*

The second approach consists of a factored statistical model whose parameters are estimated from the Wizard-of-Oz data set collected for the experiment. We assume that both the transition and utility models of the domain are initially unknown. In both this approach and Approach 3, the action selected by the robot simply corresponds to the action yielding the highest utility in the current dialogue state.

The transition model is itself factored into three models: a task completion model (encoding the probability that the current instruction is fulfilled by the last system action), a user goal model (encoding the probability of a given instruction given the previous instruction, its completion status, and the physical location of the objects), and a user action model (encoding the probability of the next user dialogue act given the intended instruction to execute and the last system action). The utility model expresses the utility of system actions given the user intention, last user dialogue act, robot's motion status, and physical objects currently perceived/carried by the robot. The resulting model is shown in Fig. 8(a). Upon the reception of new observations (coming from the speech recogniser or from other components such as the vision modules), the dialogue state is first updated on the basis of the transition model, and the utility model is then applied to select the next action (if any).

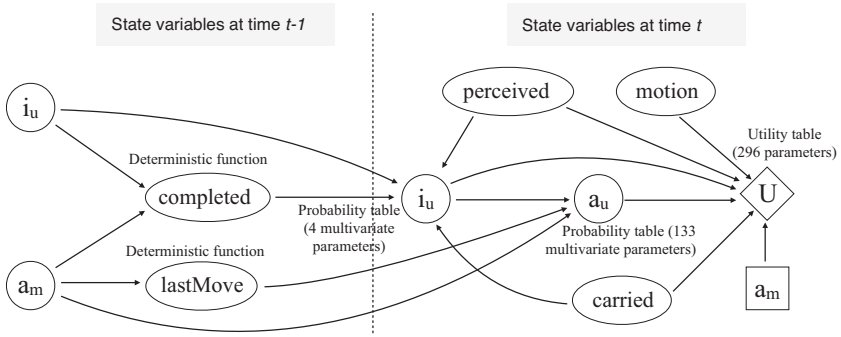
A total of 433 unknown parameters (including both univariate and multivariate parameters) are associated to the factored model. The parameters of categorical distributions are encoded by Dirichlet distributions initialised with weakly informative priors, while the utility parameters are encoded by uniform distributions. The parameter estimation follows the Bayesian learning procedure described in Section 4.1.

*Approach 3: Rule-structured model*

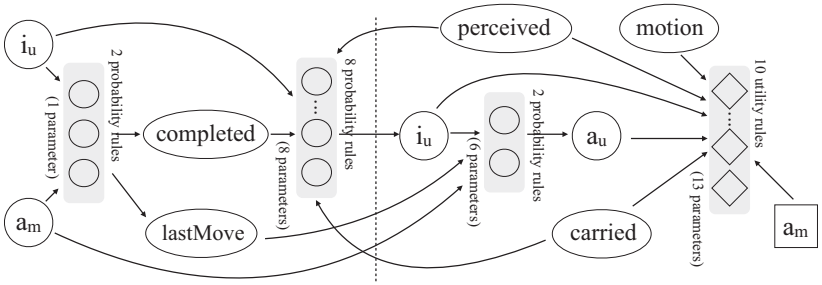
The third approach relies on probabilistic rules. As in Approach 2, the domain models are decomposed into a task completion model, a user goal model, a user action model, and a utility model. However, in contrast to the previous approach, the domain models are here structured through probabilistic rules instead of traditional distributions. The graphical model resulting from the instantiation of these rules is shown in Fig. 8(b).

The transition model is represented by a total of 12 probability rules associated with 15 probability parameters (whose prior distributions are expressed as Dirichlets), while the utility model is represented by 10 utility rules and 13 associated parameters (with uniform prior distributions).[7] The rule parameters are again estimated via Bayesian inference on the Wizard-of-Oz dataset, following the procedure outlined in Section 4.1. The only difference between

[7] The complete domain specification is available in Lison, 2014, pp. 204–212.




(a) Factored probabilistic model (Approach 2)




(b) Rule-structured probabilistic model (Approach 3)

Fig. 8. Probabilistic graphical models employed in Approach 2 (top) and Approach 3 (bottom). The variable $a_m$ is the last system action, $i_u$ represents the user intention (the variable at time $t-1$ is the intention before the execution of the system action, and the one at time $t$ the intention after it), *lastMove* the robot's last physical movement, *completed* expresses whether the user intention has been fulfilled by the last system action, $a_u$ is the new user dialogue act, *perceived* represents the objects perceived by the robot in the visual scene (if any), *carried* represents the object currently carried by the robot (if any), *motion* indicates whether the robot is currently executing a physical movement, and the action variable $a_m$ at time $t$ is the next system action. The variables $a_u$, *perceived*, *carried* and *motion* correspond to (possibly noisy) observations that are regularly updated by external components (language understanding, vision modules, etc.).

the two approaches lies therefore in the internal representation of the domain models, namely factored distributions for Approach 2 and probabilistic rules for Approach 3.

### 6.3. Learning curves

The Wizard-of-Oz dataset was employed to tune the parameters of the dialogue management strategies. Fig. 9 shows typical learning curves for the three strategies. To generate the learning curves, the Wizard-of-Oz data set was divided into a training set of 9 interactions (summing to 770 system turns) and a held-out test set with one single interaction containing a total of 71 system turns. Based on this division, we measured the degree of agreement between the actions selected by the system (on the basis of its model parameters) and the ones selected by the wizard in the test set. This measurement is repeated at regular intervals during the parameter estimation process. As the finite-state approach is entirely hand-crafted and does not include any learnable parameters, its agreement with the wizard actions remains constant.

We can observe from the learning curves that both statistical approaches (Approach 2 and 3) improve their agreement as they process more and more training samples, but do so at different learning speeds. While the rule-structured model is able to converge to a high-quality dialogue policy after observing only a fraction of the training data, the factored statistical model converges at a slower rate due to its larger set of parameters and weaker generalisation capacity. It should, however, be noted that system actions that differ from the wizard choices are not necessarily wrong or inappropriate, as they may reflect different but legitimate conversational strategies. The degree of agreement between the model and the wizard examples is nevertheless a good indication of the model's ability to capture the dynamics of the interaction as well as the trade-offs of the action selection process.

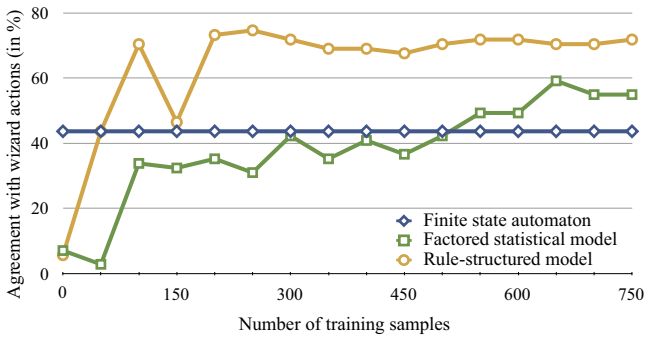



Fig. 9. Learning curves on the Wizard-of-Oz data set. The figure shows how the agreement between the system and the actual wizard actions on the held-out dialogue evolves as a function of the number of processed training examples.

### 6.4. Experimental setup

In order to assess the performance of the three dialogue management approaches, we conducted a total of $3 \times 37$ user trials. The experiment was carried out with 37 participants (16 males and 21 females). The average age of the participants was 30.6 years (standard deviation: 7.8). The participants were recruited amongst students and employees at the University of Oslo. All participants were non-native speakers of English. The participants were also asked to fill out a short survey about their age, gender, as well as their prior expectations regarding the robot's behaviour. The user expectations were surveyed with six multiple-answer questions covering aspects such as the comprehension abilities of the robot, its ability to select appropriate reactions, its use of clarification requests, its ability to distinguish speech from non-speech in the audio signals, and the overall naturalness of the interaction. Each question allowed five alternative answers on a scale from worst to best.

Each participant carried out three distinct dialogues, one for each dialogue management approach. The order of the three approaches was shuffled to mitigate the effects of the ordering sequence on the results. Immediately after each dialogue, the participants were asked to fill out a survey to determine how they subjectively perceived the interaction. The survey was once more divided into six multiple-answer questions similar to the ones employed for the survey on prior expectations. The six survey questions are listed in Fig. 11. All interactions were captured on video, totalling about 12 h of footage. In addition to the videos, the dialogue system also recorded the logs of every user and system turns occurring in the interaction along with their associated dialogue state.

### 6.5. Quality metrics and results

The quality and efficiency of the resulting dialogues were measured using a mixture of objective and subjective metrics:

- The objective metrics O1–O9 (see Fig. 10) were computed on the basis of the logs produced by the dialogue system during the course of the interaction, and correspond to measures such as the average number of repetitions and confirmations, average number of turns, and overall dialogue duration. No measure of task completion is included since the participants could not "fail" the task (there was no time limit for the interaction).
- The subjective measures S1–S6 (see Fig. 11) reflect the answers to the six multiple-answer questions in the user survey. The quantitative results are calculated by mapping the qualitative answers to a scale from 1 to 5 (with 1 corresponding to the worst behaviour and 5 to the best) and averaging the results over all participants.

Table 1 summarises the results on the complete set of metrics and indicates the statistical significance of each observed difference. The table indicates the $p$-values for two distinct test statistics: single-factor ANOVA (with the selected dialogue manager representing the factor) and Bonferroni-corrected Student's paired $t$-tests between the best and second-best dialogue manager among the three approaches.

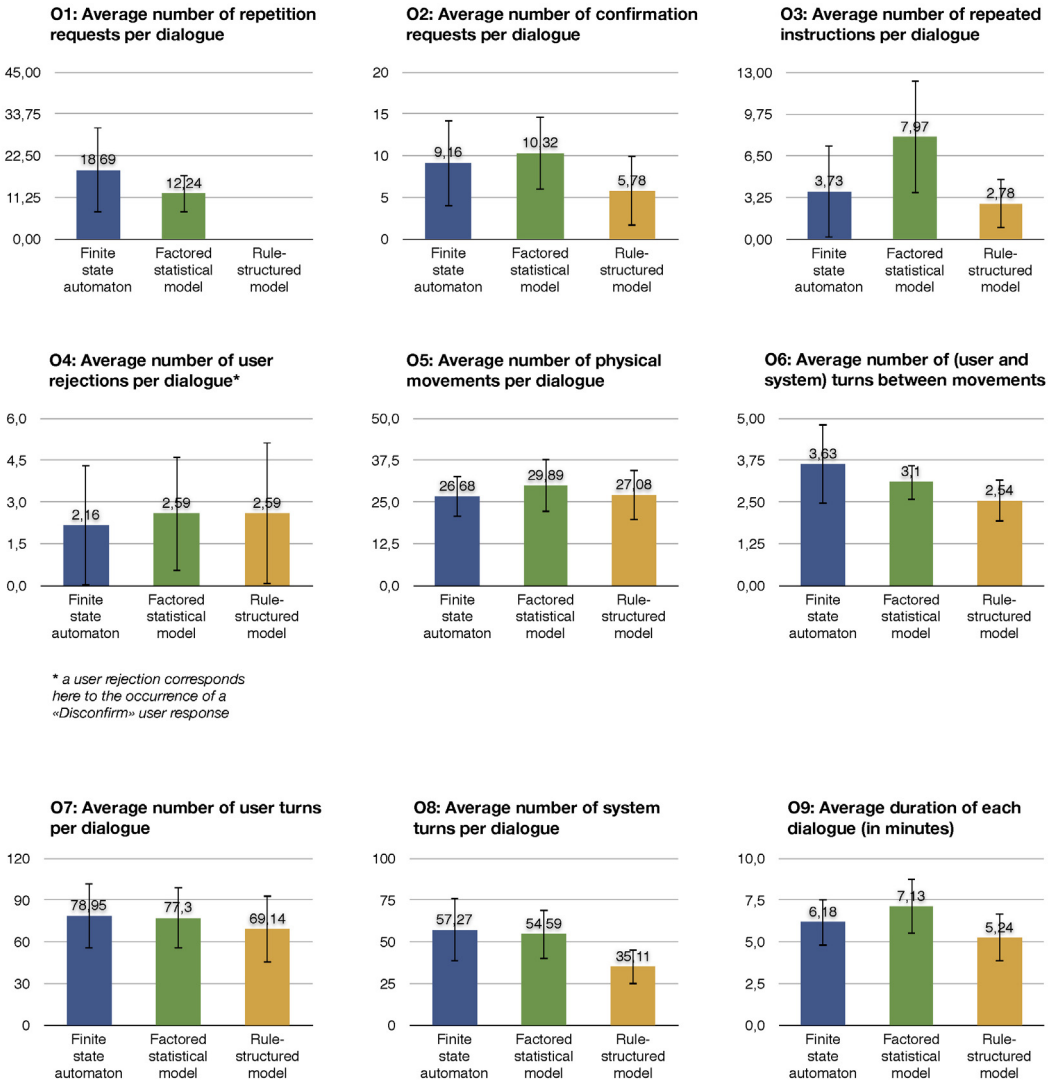



Fig. 10. Objective metrics O1-O9 extracted from the collected dialogues for each dialogue strategy: finite-state automaton (Approach 1), factored statistical model (Approach 2) or rule-structured model (Approach 3).

Table 1 shows that 7 of the 15 metrics demonstrate statistically significant differences for the rule-structured model, while 6 metrics show higher averages for the rule-structured model but without achieving statistical significance. Two objective metrics yield slightly better results for the finite-state approach, although both without statistical significance. It is also interesting to contrast the subjective results with the initial user expectations obtained from the survey conducted before the start of the experiment (second column in Table 1). For all six questions, the system behaviour for the rule-structured approach exceeded the initial expectations of the user, while the two baseline models exhibit worse-than-expected performance on most of the subjective metrics.

### 6.6. Error analysis and interpretation

The empirical results obtained from the user trials corroborate the central claim of this article, namely that probabilistic rules constitute a viable and competitive alternative to classical dialogue modelling techniques such as finite-state

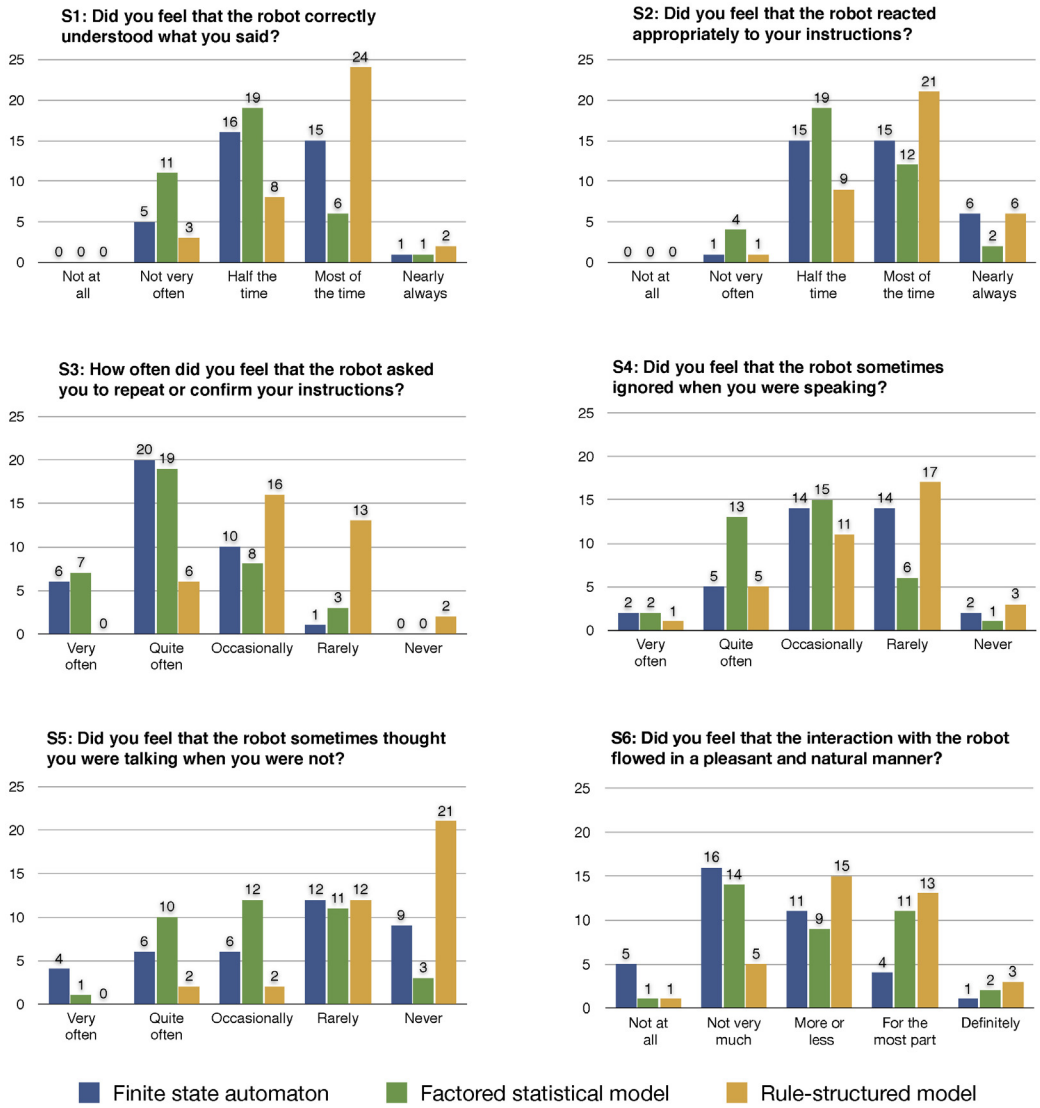



Fig. 11. Survey responses obtained from the participants after each interaction, depending on the selected dialogue manager. The *Y* axis corresponds to the number of user responses (out of 37 participants).

automata and factored probabilistic models. Through a follow-up error analysis, we found this difference to be imputable to several factors.

Although the finite-state automaton functioned well in most trials, it often required multiple repetitions to reach sufficient confidence regarding the instruction to perform, due to the noisy outputs of the speech recogniser. In comparison, the rule-structured model was able to accumulate evidence from several information sources (including the prior probabilities of the various instructions given the context) thanks to its probabilistic representation of the user intention. This possibility to accumulate evidence and exploit contextual knowledge enables the rule-structured model to focus on the right instruction faster than the hand-crafted model.

The factored statistical model also operated well in the most common conversational situations, but was frequently led astray when encountering events that had not been observed in the Wizard-of-Oz training set. In particular, although the user action model offered reasonable estimates for the most prevalent user intentions and dialogue acts, this was not the case for less common instructions which had few or no occurrences in the training set. This disparity in the estimated distributions often resulted in the statistical dialogue manager getting stuck in unseen dialogue states and

Table 1
Empirical results for the user evaluation with 37 participants, based on a set of 15 objective (O) and subjective (S) metrics. The best results for each metric are shown in bold, and the * symbol indicates results that outperform the two other approaches with significance $\alpha = 0.05$ and Bonferroni correction. See Figs. 10 and 11 for a description of each individual metric.

| Metrics | Prior expectations | Approaches | | | $p$-values one-factor ANOVA | $p$-values 1st vs. 2nd ($t$-test) |
|---|---|---|---|---|---|---|
| | | Finite-state automaton | Factored statistical model | Rule-structured model | | |
| O1. | | 18.68 | 12.24 | **0**$^*$ | $9 \times 10^{-19}$ | $1 \times 10^{-16}$ |
| O2. | | 9.16 | 10.32 | **5.78**$^*$ | $1.7 \times 10^{-4}$ | 0.001 |
| O3. | | 3.73 | 7.97 | **2.78**$^*$ | $3.8 \times 10^{-9}$ | 0.18 |
| O4. | | **2.16** | 2.59 | 2.59$^*$ | 0.65 | 0.33 |
| O5. | | **26.68** | 29.89 | 27.08$^*$ | 0.13 | 0.80 |
| O6. | | 3.63 | 3.10 | **2.54**$^*$ | $4. \times 10^{-4}$ | $2.2 \times 10^{-4}$ |
| O7. | | 78.95 | 77.30 | **69.14**$^*$ | 0.17 | 0.11 |
| O8. | | 57.27 | 54.59 | **35.11**$^*$ | $4.4 \times 10^{-9}$ | $5.6 \times 10^{-8}$ |
| O9. | | 6:18 | 7:13 | **5:24**$^*$ | $1.4 \times 10^{-4}$ | 0.02 |
| S1. | 3.22 | 3.32 | 2.92 | **3.68** | $1.3 \times 10^{-4}$ | 0.03 |
| S2. | 3.35 | 3.70 | 3.32 | **3.86**$^*$ | $7.6 \times 10^{-3}$ | 0.23 |
| S3. | 2.68 | 2.16 | 2.19 | **3.30**$^*$ | $1.7 \times 10^{-9}$ | $4.7 \times 10^{-7}$ |
| S4. | 2.89 | 3.24 | 2.76 | **3.43**$^*$ | $6.7 \times 10^{-3}$ | 0.21 |
| S5. | 3.81 | 3.43 | 3.14 | **4.41**$^*$ | $3.4 \times 10^{-6}$ | $4.7 \times 10^{-5}$ |
| S6. | 2.92 | 2.97 | 2.46 | **3.32** | $8.6 \times 10^{-4}$ | 0.03 |

selecting suboptimal actions. Although it was trained on the exact same data, the rule-structured approach did not suffer from this data sparsity due to its more powerful generalisation abilities.

Finally, the parameters estimated from the Wizard-of-Oz examples for the rule-structured model systematically assigned higher utilities to confirmation requests ("*should I do X?*") than repetition requests ("*sorry, could you repeat?*") when faced with unclear user instructions, as evidenced by the metric O1 in Fig. 10. The wizard indeed often preferred to employ confirmation requests whenever possible. This proved to be an excellent conversational strategy for the task, as participants perceived these confirmation requests to be more natural and informative than repetitions. By contrast, the two baseline approaches relied more heavily on repetition requests to elicit the user instructions, a strategy that many users found irritating.

The results obtained for the two baselines are necessarily contingent on the design choices described in Section 6.2. Although these design choices follow standard dialogue modelling techniques, these strategies could certainly be further improved through the use of more advanced estimation methods and parameter tuning. However, the key problem faced by Approach 2 is the scarcity of available training data (less than 1000 Wizard-of-Oz turns) rather than the performance of the parameter estimation method. We speculate therefore that, in the absence of additional constraints on the size of the parameter space, the type of statistical technique employed in the parameter estimation is unlikely to lead to major changes in the quality metrics for this evaluation setup. The empirical results described in this section are best interpreted as a comparison between approaches featuring the same level of sophistication in their domain representation, but differing from one another along two dimensions:

1. The automaton and the rule-structured model share the same expert domain knowledge and mappings between user instructions and corresponding system actions. However, they diverge in their account of uncertainty: while the automaton relies on confidence thresholds, the rule-structured model employs probabilistic reasoning to infer the most likely instruction given the observed inputs, prior likelihoods and external context.
2. Analogously, the factored statistical model and the rule-structured model share the state variables, conditional dependencies and training data. However, they differ in the amount of internal domain structure introduced into the probabilistic model, and therefore in the numbers of parameters to estimate.

## 7. Discussion and related work

The use of structural knowledge in probabilistic models is a recurring theme in the fields of reinforcement learning (Hauskrecht et al., 1998; Pineau, 2004; Kersting and Raedt, 2004; Lang and Toussaint, 2010; van Otterlo, 2012), and

statistical relational learning (Jaeger, 2001; Richardson and Domingos, 2006; Getoor and Taskar, 2007). This structure may take a hierarchical or relational form. As in the probabilistic rules presented in this article, most of these frameworks rely on the use of expressive representations (which are often based on first-order logic) serving as templates for the generation of classical probabilistic models. While the formalisation presented in this article and the aforementioned approaches share many similar characteristics, they also differ in the following aspects:

- Most importantly, probabilistic rules are designed to operate under partially observable settings, since state uncertainty is a pervasive and unavoidable aspect of verbal interactions. By contrast, most previous work on relational probabilistic models is limited to fully observable environments, with the exception of some preliminary theoretical studies by Wang and Khardon (2010) and Sanner and Kersting (2010).
- The formalism is also primarily tailored for dialogue management and seek to capture dialogue domains by striking a balance between propositional and first-order logic. The formalism deliberately eschews the complexity of full-scale first-order probabilistic inference to ensure that the domain models can be applied under real-time constraints.
- Finally, the presented framework posits that the *if*. . .*then*. . .*else* skeletons of probabilistic rules are best encoded by the system designers based on their expert knowledge of the domain, while the parameters of the rules can be estimated empirically. We therefore exclude the problem of structure learning from the scope of this framework, as opposed to previous work in machine learning in which the structure and parameters of the rules are learned simultaneously (Pasula et al., 2007; Kok and Domingos, 2009).

Probabilistic rules bear similarities with planning description languages such as the Planning Domain Definition Language (PDDL, see McDermott et al., 1998) and its probabilistic extension, the Probabilistic Planning Domain Definition Language (PPDDL, see Younes and Littman, 2004). These languages are structured through action schemas that specify how (parametrised) actions can yield particular effects under various conditions. Contrary to the probabilistic rules presented in this article, these planning languages are, however, typically restricted to fully observable settings and require the domain to be completely specified in advance, including all probability and utility parameters employed in the action schemas. A relational extension of PDDL, named RDDL, has been introduced in recent planning competitions (Sanner, 2010). The rule learning techniques developed by Pasula et al. (2007) for the estimation of transition functions based on noisy indeterministic deictic rules are directly related to our approach, as is the recent work of Lang and Toussaint (2010) on probabilistic noisy planning rules. Both frameworks define conditions associated with probabilistic distributions over effects, but are again restricted to fully observable domains.

The reinforcement learning approach outlined in Section 4.2 builds on previous work in the field of model-based reinforcement learning. Png et al. (2012) describe a Bayes-Adaptive POMDP framework and illustrate its use in simulated interactions. Doshi and Roy (2008) present a related POMDP framework with model uncertainty combined with active learning. Action selection is formalised in their paper by sampling possible POMDP models and extracting a solution for each sample. A similar strategy is employed by Atrash and Pineau (2009). Finally, Chinaei and Chaib-draa (2012) and Chinaei et al. (2012) demonstrate the estimation of observation and reward models for dialogue POMDPs in a healthcare application.

Most structural approaches to dialogue management rely on a clear-cut task decomposition into goals and sub-goals (Allen et al., 2000; Steedman and Petrick, 2007; Bohus and Rudnicky, 2009), where the completion of each goal is assumed to be fully observable, discarding any remaining uncertainty. The processing workflow of the framework is strongly inspired by information state approaches to dialogue management (Larsson and Traum, 2000; Bos et al., 2003), which are also based on a shared state representation that is updated according to a rich repository of rules. Ginzburg (2012) also models conversational phenomena by way of update operations that are encoded with rules mapping conditions to effects. However, contrary to the framework presented here, the rules employed in these approaches are generally deterministic and do not include learnable parameters.

The literature on dialogue policy optimisation with reinforcement learning also contains several approaches dedicated to dimensionality reduction for large state–action spaces, such as function approximation (Henderson et al., 2008), hierarchical reinforcement learning (Cuayáhuitl et al., 2010) and summary POMDPs (Young et al., 2010). These approaches are, however, often engineered towards slot-filling applications and are difficult to transfer to other dialogue domains. The idea of state space partitioning, implemented here via logical conditions, has also been explored in recent papers [see e.g. Williams, 2010]. Crook and Lemon (2010) explored the introduction of complex user goal states including disjunction and negation operators. Cuayáhuitl (2011) describes a policy optimisation approach based

on logic-based representations of the state–action space for relational MDPs. The main difference to our approach lies in his reduction of the belief state to fully observable variables whereas we retain the partial observability associated with each variable. Mehta et al. (2010) and Raux and Ma (2011) demonstrated how tree-structured Bayesian networks called probabilistic ontology trees can improve belief tracking performance. Heeman (2007) and Williams (2008) have investigated how to combine dialogue policies optimised via reinforcement learning with expert knowledge encoded through information-state update rules and finite-state constraints. This expert knowledge is, however, reduced in their work to an external filtering mechanism, while the formalism of probabilistic rules integrates it into the very structure of the statistical model.

Finally, recent work has investigated the use of reinforcement learning techniques based on Gaussian processes for dialogue policy optimisation (Gašić et al., 2013). Gaussian processes can efficiently estimate $Q$-values from direct interactions using a variant of the SARSA algorithm. Gaussian Processes do not necessitate any domain-specific expertise, but are conversely unable to incorporate prior knowledge and domain constraints into the dialogue models. In other words, dialogue policy optimisation with Gaussian Processes (or similar non-parametric techniques) are likely to work best when large amounts of interaction data is available, while probabilistic rules are most advantageous when interaction data is scarce but the system designer has a good grasp of the domain structure.

## 8. Conclusion

This article presented a hybrid approach to dialogue management that combines statistical modelling with expert domain knowledge. As in existing POMDP approaches to dialogue management, the dialogue state is represented as a Bayesian network that is regularly updated with observations and used to derive high-utility system actions. The internal models of the dialogue domain are, however, represented via *probabilistic rules*. At runtime, the probabilistic rules for the domain are instantiated as latent nodes in order to update the current dialogue state with new observations and perform action selection. The probabilistic rules may include learnable parameters that can be efficiently estimated through Bayesian inference based on collected dialogue data. We outlined two alternative estimation methods for these rule parameters, respectively based on supervised learning on Wizard-of-Oz data and reinforcement learning from direct interactions.

The formalism enables the system designer to express the internal structure of a given dialogue domain in a compact set of rules. This methodology offers important benefits for dialogue management. The first benefit is a substantial reduction of the number of parameters associated with the model, thereby allowing probabilistic rules to be optimised from much smaller amounts of data. This property is critical for many application domains, as high-quality, in-domain interaction data is often scarce and difficult to acquire. The formalism is in our view particularly well-suited to model "open-ended" dialogue domains such as the ones often found in human–robot interaction, personal assistants or tutoring systems. These domains must often deal with state–action spaces that include multiple tasks to perform, complex user models, and a rich, dynamic environment. They are consequently difficult to capture with a user simulator, and typically have very limited amounts of available in-domain data. Another benefit of probabilistic rules is their improved readability for system designers, who are able to exploit powerful abstractions to encode their prior knowledge of the application domain.

Future work will extend this formalism in several directions. The first line of work is to optimise rule parameters from live interactions based on human-generated feedback instead of Wizard-of-Oz data, using techniques such as interactive shaping (Knox, 2012) to infer posterior parameter distributions. Another research direction is to estimate rule parameters using a combination of supervised and reinforcement learning. One promising strategy, already investigated in previous work on dialogue optimisation (see e.g. Williams and Young, 2003; Rieser and Lemon, 2006; Henderson et al., 2008), is to derive initial estimates via supervised learning and subsequently refine them through reinforcement learning. Such a combination of approaches would be facilitated in our framework by the fact that both learning techniques are already grounded in the same theoretical foundations, namely Bayesian inference.

Finally, the framework should be evaluated in the context of more sophisticated dialogue domains. While the human–robot interaction scenario described in Section 6 presented a number of non-trivial challenges to address (notably in regard to the high proportion of speech recognition errors and the necessity to perceive and act upon a real physical environment), it remained restricted to a relatively simple task. Scaling up the framework to domains with larger state–action spaces and more elaborate dialogue dynamics would allow us to cast a new light on the expressive power of the formalism and its suitability for practical dialogue applications.

## Acknowledgements

The research presented in this article was funded by a Ph.D. Research Fellowship from the University of Oslo and a Post-doctoral Research Grant from the Norwegian Research Council. The author wishes to thank Stephan Oepen, Erik Velldal, Geert-Jan M. Kruijff, Kristiina Jokinen, Verena Rieser and Andrei Popescu-Belis for many useful comments.

## Appendix A. Video footage

A video footage of the human-robot interaction scenario presented in Section 6 can be found, in the online version, at http://dx.doi.org/10.1016/j.csl.2015.01.001.

## References

Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., Stent, A., 2000. An architecture for a generic dialogue shell. Nat. Lang. Eng. 6, 213–228.

Atrash, A., Pineau, J.,2009. A Bayesian reinforcement learning approach for customizing human–robot interfaces. In: Proceedings of the 13th International Conference on Intelligent User Interfaces. Sanibel Island, USA, pp. 355–360.

Billard, A., Calinon, S., Dillmann, R., Schaal, S., 2008. Robot programming by demonstration. In: Siciliano, B., Khatib, O. (Eds.), Springer Handbook of Robotics. Springer, Secausus, USA, pp. 1371–1394.

Bohus, D., Rudnicky, A., 2009. The RavenClaw dialog management framework: architecture and systems. Comput. Speech Lang. 23 (3), 332–361.

Bos, J., Klein, E., Lemon, O., Oka, T.,2003. DIPPER: description and formalisation of an information-state update dialogue system architecture. In: Proceedings of the 4th SIGDIAL Workshop on Discourse and Dialogue. Sapporo, Japan, pp. 115–124.

Bui, T.H., Poel, M., Nijholt, A., Zwiers, J., 2009. A tractable hybrid DDN-POMDP approach to affective dialogue modeling for probabilistic frame-based dialogue systems. Nat. Lang. Eng. 15 (2), 273–307.

Chinaei, H.R., Chaib-draa, B.,2012. An inverse reinforcement learning algorithm for partially observable domains with application on healthcare dialogue management. In: Proceedings of the 11th International Conference on Machine Learning and Applications. Boca Raton, USA, pp. 144–149.

Chinaei, H.R., Chaib-draa, B., Lamontagne, L.,2012. Learning observation models for dialogue POMDPs. In: Proceedings of the 25th Canadian Conference on Artificial Intelligence. Toronto, Canada, pp. 280–286.

Corkill, D.D., 1989. Design alternatives for parallel and distributed blackboard systems. In: Blackboard Architectures and Applications. Academic Press, San Diego, USA, pp. 99–136.

Crook, P.A., Lemon, O.,2010. Representing uncertainty about complex user goals in statistical dialogue systems. In: Proceedings of the 12th Annual SIGDIAL Meeting on Discourse and Dialogue. Tokyo, Japan, pp. 209–212.

Cuayáhuitl, H.,2011. Learning dialogue agents with Bayesian relational state representations. In: Proceedings of the IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems. Barcelona, Spain, pp. 9–15.

Cuayáhuitl, H., Renals, S., Lemon, O., Shimodaira, H., 2010. Evaluation of a hierarchical reinforcement learning spoken dialogue system. Comput. Speech Lang. 24, 395–429.

Daubigney, L., Geist, M., Pietquin, O., 2012. Off-policy learning in large-scale POMDP-based dialogue systems. In: Proceedings of the 37th IEEE International Conference on Acoustics, Speech, and Signal Processing, Kyoto, Japan, pp. 4989–4992.

Doshi, F., Roy, N., 2008. Spoken language interaction with model uncertainty: an adaptive human–robot interaction system. Connect. Sci. 20 (4), 299–318.

Frampton, M., Lemon, O., 2009. Recent research advances in reinforcement learning in spoken dialogue systems. Knowl. Eng. Rev. 24 (4), 375–408.

Fung, R.M., Chang, K.-C.,1989. Weighing and integrating evidence for stochastic simulation in Bayesian networks. In: Proceedings of the 5th Conference on Uncertainty in Artificial Intelligence. Windsor, Canada, pp. 209–220.

Gašić, M., Breslin, C., Henderson, M., Kim, D., Szummer, M., Thomson, B., Tsiakoulis, P., Young, S.,2013. On-line policy optimisation of Bayesian spoken dialogue systems via human interaction. In: Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing. Vancouver, Canada.

Gašić, M., Jurčí ček, F., Thomson, B., Yu, K., Young, S., 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In: IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 312–317.

Getoor, L., Taskar, B., 2007. Introduction to Statistical Relational Learning. MIT Press, Cambridge, USA.

Ginzburg, J., 2012. The Interactive Stance. Oxford University Press, Oxford, UK.

Hauskrecht, M., Meuleau, N., Kaelbling, L.P., Dean, T., Boutilier, C.,1998. Hierarchical solution of Markov decision processes using macro-actions. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. Madison, USA, pp. 220–229.

Heeman, P.,2007. Combining reinforcement learning with information-state update rules. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics. Rochester, USA, pp. 268–275.

Henderson, J., Lemon, O., Georgila, K., 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. Comput. Linguist. 34, 487–511.

Jaeger, M., 2001. Complex probabilistic modeling with recursive relational Bayesian networks. Ann. Math. Artif. Intell. 32 (1–4), 179–220.

Jokinen, K., 2009. Constructive Dialogue Modelling: Speech Interaction and Rational Agents. Wiley, Chichester, UK.

Kersting, K., Raedt, L.D.,2004. Logical Markov decision programs and the convergence of logical TD(lambda). In: Proceedings of the 14th International Conference on Inductive Logic Programming. Porto, Portugal, pp. 180–197.
Knox, W.B., 2012. Learning from Human-generated Reward. University of Texas, Austin (Ph.D. thesis).
Kok, S., Domingos, P., 2009. Learning Markov logic network structure via hypergraph lifting. In: Proceedings of the 26th International Conference on Machine Learning, Montréal, Canada, pp. 505–512.
Koller, D., Friedman, N., 2009. Probabilistic Graphical Models: Principles and Techniques. MIT Press, Cambridge, USA.
Lang, T., Toussaint, M., 2010. Planning with noisy probabilistic relational rules. J. Artif. Intell. Res. 39, 1–49.
Larsson, S., 2002. Issue-based Dialogue Management. University of Göteborg (Ph.D. thesis).
Larsson, S., Traum, D.R., 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. Nat. Lang. Eng. 6 (3–4), 323–340.
Levin, E., Pieraccini, R., Eckert, W., 2000. A stochastic model of human–machine interaction for learning dialog strategies. IEEE Trans. Speech Audio Process. 8 (1), 11–23.
Li, L., Walsh, T.J., Littman, M.L.,2006. Towards a unified theory of state abstraction for MDPs. In: Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics. Fort Lauderdale, USA.
Lison, P.,2013. Model-based Bayesian reinforcement learning for dialogue management. In: Proceedings of the 14th Annual Conference of the International Speech Communication Association. Lyon, France.
Lison, P., 2014. Structured Probabilistic Modelling for Dialogue Management. University of Oslo (Ph.D. thesis).
McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D., 1998. PDDL – the planning domain definition language. Tech. Rep. CVC TR98003/DCS TR1165. Yale Center for Computational Vision and Control.
Mehta, N., Gupta, R., Raux, A., Ramachandran, D., Krawczyk, S.,2010. Probabilistic ontology trees for belief tracking in dialog systems. In: Proceedings of the 11th Annual SIGDIAL Meeting on Discourse and Dialogue. Tokyo, Japan, pp. 37–46.
Pasula, H.M., Zettlemoyer, L.S., Kaelbling, L.P., 2007. Learning symbolic models of stochastic domains. J. Artif. Intell. Res. 29, 309–352.
Pietquin, O., 2008. Optimising spoken dialogue strategies within the reinforcement learning paradigm. In: Reinforcement Learning, Theory and Applications. I-Tech Education and Publishing, Vienna, Austria, pp. 239–256.
Pineau, J., 2004. Tractable Planning Under Uncertainty: Exploiting Structure. Carnegie Mellon University (Ph.D. thesis).
Png, S., Pineau, J., Chaib-draa, B., 2012. Building adaptive dialogue systems via Bayes-Adaptive POMDPs. IEEE J. Sel. Top. Signal Process. 6 (8), 917–927.
Raux, A., Ma, Y.,2011. Efficient probabilistic tracking of user goal and dialog history for spoken dialog systems. In: Proceedings of the 12th Annual Conference of the International Speech Communication Association. Florence, Italy, pp. 801–804.
Richardson, M., Domingos, P., 2006. Markov logic networks. Mach. Learn. 62, 107–136.
Rieser, V., Lemon, O.,2006. Using logistic regression to initialise reinforcement-learning-based systems. In: Proceedings of the IEEE Spoken Language Technology Workshop. Palm Beach, Aruba, pp. 190–193.
Rieser, V., Lemon, O., 2011. Reinforcement Learning for Adaptive Dialogue Systems. Springer, Berlin/Heidelberg, Germany.
Roy, N., Pineau, J., Thrun, S.,2000. Spoken dialogue management using probabilistic reasoning. In: Proceedings of the 38th Meeting of the Association for Computational Linguistics. Hong Kong, China, pp. 93–100.
Sanner, S., 2010. Relational Dynamic Influence Diagram Language (RDDL): Language Description.
Sanner, S., Kersting, K., 2010. Symbolic dynamic programming for first-order POMDPs. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence, Atlanta, USA.
Steedman, M., Petrick, R.P.A., 2007. Planning dialog actions. In: Proceedings of the 8th SIGDIAL Workshop on Discourse and Dialogue, Antwerp, Belgium, pp. 265–272.
Thomson, B., Young, S., 2010. Bayesian update of dialogue state: a POMDP framework for spoken dialogue systems. Comput. Speech Lang. 24, 562–588.
van Otterlo, M., 2012. Solving relational and first-order logical Markov decision processes: a survey. In: Reinforcement Learning. Adaptation, Learning, and Optimization, vol. 12. Springer, Berlin/Heidelberg, Germany, pp. 253–292.
Wang, C., Khardon, R., 2010. Relational partially observable MDPs. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence, Atlanta, USA, pp. 1153–1158.
Williams, J.D., 2008. The best of both worlds: unifying conventional dialog systems and POMDPs. In: Proceedings of the 9th Annual Conference of the International Speech Communication Association, Brisbane, Australia.
Williams, J.D., 2010. Incremental partition recombination for efficient tracking of multiple dialog states. In: Proceedings of the 35th IEEE International Conference on Acoustics, Speech, and Signal Processing, Dallas, USA, pp. 5382–5385.
Williams, J.D., Young, S., 2003. Using Wizard-of-Oz simulations to bootstrap reinforcement-learning based dialog management systems. In: Proceedings of the 4th SIGDIAL Workshop on Discourse and Dialogue, Sapporo, Japan.
Younes, H.L.S., Littman, M.L., 2004. PPDDL1.0: the language for the probabilistic part of IPC-4. In: Proceedings of the 4th International Planning Competition, Whistler, Canada.
Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., Yu, K., 2010. The hidden information state model: a practical framework for POMDP-based spoken dialogue management. Comput. Speech Lang. 24, 150–174.