Declarative Design of Spoken Dialogue Systems with Probabilistic Rules

Pierre Lison
Department of Informatics
University of Oslo, Norway
plison@ifi.uio.no

Abstract

Spoken dialogue systems are instantiated in complex architectures comprising multiple interconnected components. These architectures often take the form of pipelines whose components are essentially black-boxes developed and optimised separately, using ad-hoc specification formats for their inputs and outputs, domain models and parameters.

We present in this paper an alternative modelling approach, in which the dialogue processing steps (from understanding to management and to generation) are all declaratively specified using the same underlying formalism. The formalism is based on *probabilistic rules* operating on a shared belief state. These rules are expressed as structured mapping between state variables and provide a compact, probabilistic encoding for the dialogue processing models. We argue that this declarative approach yields several advantages in terms of transparency, domain-portability and adaptivity over traditional black-box architectures. We also describe the implementation and validation of this approach in an integrated architecture for human-robot interaction.

1 Introduction

Spoken dialogue systems typically rely on complex pipeline architectures, including components such as speech recognition, semantic parsing, dialogue act classification, dialogue management, sentence planning, surface realisation and speech synthesis, in addition to extra-linguistic modules for e.g. situation awareness or the execution of physical actions, see e.g. (Bos et al., 2003; Bohus et al., 2007; Kruijff et al., 2007). In many cases, these components are developed separately and rely on dedicated representation formats for their inputs and output variables, internal models and parameters.

For instance, a dialogue act classifier might take as input a N-Best list of recognition hypotheses, and outputs the corresponding dialogue act(s), using a set of shallow patterns as internal model to relate the input to the output. Similarly, a dialogue manager takes a given dialogue state as input, and outputs the optimal action (if any) to perform in such state based on a specific planning procedure.

These representation formats are unfortunately rarely compatible with one another, which makes it difficult to derive a semantic interpretation for the dialogue state as a whole (in terms e.g. of a joint probability distribution). Moreover, much of the task knowledge is typically encoded in procedural form within the component workflow, reducing the system portability to other domains due to the necessity of reprogramming some of the modules.

We present in this paper an alternative approach to the specification and optimisation of the various models used in a dialogue system architecture. The key idea is to declaratively specify the task-specific models using a shared, generic representation formalism, and strip down the system architecture to a small set of core algorithms for updating the dialogue state based on these models. The representation formalism we describe in this paper is based on the concept of a *probabilistic rule*. These rules are expressive enough to capture the structure for most processing tasks, from dialogue understanding to management and to generation. Moreover, they can be either manually designed or have their parameters estimated from data.

This declarative approach to the design of spo-

Proceedings of the 16th Workshop on the Semantics and Pragmatics of Dialogue, September 19–21, 2012, Paris, France.

ken dialogue systems has several advantages. The first one pertains to domain portability. Given that the dialogue architecture is essentially reduced to a generic platform for rule instantiation and inference, porting the system to a new domain only requires a rewrite or extension of the domain-specific rules, without having to reprogram a single component. It also provides a more transparent semantics for the system as a whole, since all state variables are described and related to each other in a unified and theoretically sound framework, grounded in probabilistic inference. Finally, the use of probabilistic rules enables the construction of very flexible processing pipelines, by allowing state variables to depend or influence each other in any order and direction. The system designer is thus free to combine in the same architecture both deep and shallow semantic parsers for dialogue understanding, or both reactive and deliberative policies for dialogue management.

The architecture revolves around a shared dialogue state, encoded as a Bayesian Network including all variables relevant for the interaction. The use of a Bayesian Network allows us to account for the various kinds of uncertainties arising in spoken dialogue (speech recognition errors, unknown user intentions, etc.) as well as the conditional dependencies between state variables. At runtime, this dialogue state is continuously updated via probabilistic rules. As we shall see, these rules are instantiated by extending the Bayesian network with new nodes and conditional dependencies.

We showed in our previous work how to estimate the parameters of these models given limited amounts of Wizard-of-Oz training data (Lison, 2012). The present paper builds upon this approach, but concentrates on the design and specification of these probabilistic rules for various processing tasks, leaving out the question of parameter estimation. Hence, we will simply assume through this paper that the parameters have been already assigned, either from training data or expert knowledge.

The rest of the paper is as follows. We first provide generalities about Bayesian Networks and dialogue models. We then describe our approach by defining the probabilistic rules and their use in the dialogue processing workflow. We also detail a system implemented for a human-robot interaction domain, which exploits probabilistic rules to perform tasks related to dialogue understanding, management and generation. Finally, we discuss and relate our approach to previous work, and conclude.

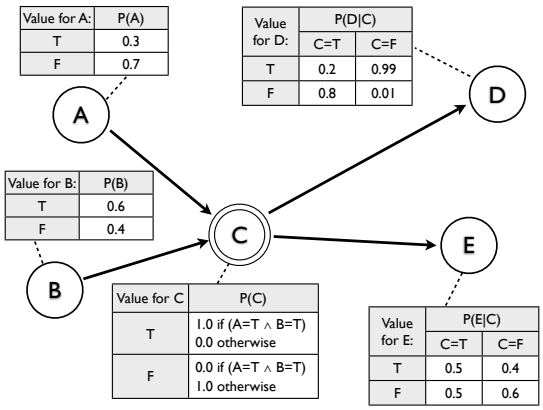

Figure 1: Bayesian network with 5 nodes. An example of query on this network is $P(A{=}\mathrm{T}|D{=}\mathrm{T}) \approx 0.18$.

2 Background

2.1 Bayesian Networks

The probabilistic models used in this paper are expressed as directed graphical models, also known as Bayesian Networks. Let $X_1 ... X_n$ denote a set of random variables. Each variable X_i is associated with a range of mutually exclusive values. In dialogue models, this range is often discrete and can be explicitly enumerated: $Val(X_i) = \{x_i^1, ..., x_i^m\}$.

A Bayesian Network defines the conditional dependencies between variables using a directed graph where each node corresponds to a variable X_i. Each edge $X_i \rightarrow X_j$ denotes a conditional dependence between the two nodes, in which case X_i is said to be a *parent* of X_j. A conditional probability distribution $P(X_i|Par(X_i))$ is associated with each node X_i, where $Par(X_i)$ denotes the parents of X_i.

Conditional probability distributions (CPDs) can be defined in various ways, from look-up tables to more advanced distributions (Koller and Friedman, 2009). Together with the directed graph, the CPDs determine the joint probability distribution $P(X_1 ... X_n)$. The network can then be used for inference by querying the distribution of a subset of variables, often given some additional evidence, as illustrated by the example in Figure 1.

2.2 Dialogue models

The dialogue state $\mathbf{s}$ can generally be decomposed into a set of state variables $\mathbf{s} = \{s_1, ...s_n\}$ (Williams and Young, 2007). Each state variable represents a relevant feature of the interaction and its context. For instance, the state variables for a human-robot interaction scenario might be composed of tasks to accomplish, the interaction history, past events, as well as objects, spatial locations and agents in the environment. A minimal dialogue state can be defined as $\mathbf{s} = \langle u_u, a_u, i_u, a_m, u_m, c \rangle$, where u_u is the last user utterance, a_u the last dialogue act, i_u the current user intention, a_m the last system act, u_m the last system utterance, and c the context.

Due to uncertainty, many variables are only partially observable. We thus encode our knowledge of the current dialogue state in a distribution $\mathbf{b}(\mathbf{s}) = P(s_1, ..., s_n)$ called the *belief state*, conveniently expressed as a Bayesian Network (Thomson and Young, 2010). This belief state $\mathbf{b}$ is regularly updated with new information. The workflow illustrated in Figure 2 can then be formalised in terms of inference steps over this belief state:

1. Upon detection of a new speech signal, the speech recogniser generates the N-best list of recognition hypotheses $\tilde{\mathbf{u}}_\mathbf{u} = P(u_u|o)$;
2. Speech understanding then estimates the most likely dialogue act(s) realised in the utterance: $\tilde{\mathbf{a}}_\mathbf{u} = P(a_u|\mathbf{b})$;
3. The user intention is updated with the new interpreted dialogue act: $\tilde{\mathbf{i}}_\mathbf{u} = P(i_u|\mathbf{b})$;
4. Based on the updated belief state, the action selection searches for the optimal system action to perform: $a^*_m = \arg\max_{a_m} Q(a_m|\mathbf{b})$;
5. The system action is then realised in an utterance u_m, which is again framed as a search for $u^*_m = \arg\max_{u_m} Q(u_m|\mathbf{b})$;

The models defined above use $P(x|\mathbf{b})$ as a notational convenience for $\sum_{\mathbf{s^i} \in Val(\mathbf{s})} P(x|\mathbf{s}=\mathbf{s^i})\mathbf{b}(\mathbf{s^i})$. The sequence above might be adapted in various ways depending on the domain. A basic reactive system might for instance ignore the user intention and directly select its actions based on the last dialogue act. Similarly, user-adaptivity might be captured via additional processing steps to estimate and exploit features related to the user model.

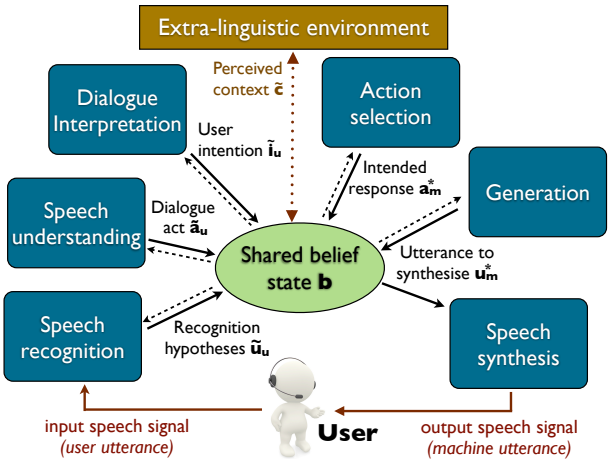

Figure 2: Dialogue system architecture schema.

3 Approach

The starting point of our approach is to express the probabilistic dialogue models described above using a compact encoding which takes advantage of the *internal structure* present in most processing tasks. This structure can take several forms:

- The probability (or utility) of a given output variable typically depends on only a small subset of input variables, although the number and identity of these variables might naturally differ. The state variable encoding the physical location of a mobile robot is for instance relevant for answering a user requesting its location, but not for responding to a greeting act.

- Moreover, the values for the dependent variables can often be grouped into a small number of *partitions* yielding similar outcomes, thereby reducing the dimensionality of the problem. The partitions can be expressed via logical conditions on the variable values.

Probabilistic rules provide a general framework for expressing this internal structure. The rules express the model distribution in terms of structured mappings between input and output variables. At runtime, these rules are then applied on the belief state, thereby extending the Bayesian Network with new nodes and conditional dependencies. This updated Bayesian Network can then be directly used

for inference, e.g. to compute the marginal distribution of a particular variable or to search for the optimal action to perform. The probabilistic rules thus function as high-level *templates* for the incremental construction of a classical probabilistic model.

3.1 Definitions

Rules can be of two possible types:

1. *probability rules*, which define probability models of the form $P(\mathbf{X}|\mathbf{Y})$, where $\mathbf{X}$ and $\mathbf{Y}$ both denote arbitrary sets of state variables;

2. *utility rules*, defining utility models of the form $Q(\mathbf{X}, \mathbf{A})$, where $\mathbf{X}$ represent a set of state variables, and $\mathbf{A}$ a set of action variables. The rule defines here the utility of a particular action sequence in $\mathbf{A}$ given the state defined by $\mathbf{X}$.

A rule is essentially defined as a condition-effect mapping, where each condition is mapped to a set of alternative effects. Depending on the type of rule, each effect will be assigned to either a *probability* or a *utility* value. The list of conditions is ordered and takes the form of a "**if** ... **then** ... **else**" case expressing the probability/utility distribution of the output variables depending on the inputs.

Probability rule

Formally, a probability rule r is defined as an ordered list of cases, where each case is associated with a condition c_i as well as a distribution over stochastic effects $\{(e_i^1, p_i^1), ..., (e_i^k, p_i^k)\}$. For each stochastic effect e_i^j, we have that $p_i^j = P(e_i^j|c_i)$, where $p_i^{1...m}$ satisfy the usual probability axioms. The rule reads as such:

if (c_1) **then**
$\{P(e_1^1) = p_1^1,\ ...\ P(e_1^k) = p_1^k\}$
...
else if (c_n) **then**
$\{P(e_n^1) = p_n^1,\ ...\ P(e_n^m) = p_n^m\}$

A final **else** case is implicitly added to the bottom of the list, and holds if no other condition applies. If not overridden, the default effect associated to this last case is void – i.e. it causes no changes to the distribution over the output variables.

Utility rule

Utility rules are defined similarly. Each case specified in the rule is associated to a condition c_i and a utility distribution over possible action sequences $\{(a_i^1, q_i^1), ..., (a_i^k, q_i^k)\}$, where a_i^j is a value assignment for a set of action variables, and $q_i^j = Q(c_i, a_i^j)$. The rule reads as:

if (c_1) **then**
$\{Q(a_1^1) = q_1^1,\ ...\ Q(a_1^k) = q_1^k\}$
...
else if (c_n) **then**
$\{Q(a_n^1) = q_n^1,\ ...\ Q(a_n^m) = q_n^m\}$

The default utility value of an action is set to 0. When several rules define a utility value for the same action, these utilities are *summed*.

Conditions

For both rule types, the conditions are expressed as logical formulae grounded in the input variables. They can be arbitrarily complex formulae connected by conjunctive, disjunctive and negation operators. The conditions on the input variables can be seen as providing a compact partition of the state space to mitigate the dimensionality curse. Without this partitioning in alternative conditions, a rule ranging over m variables each of size n would need to enumerate n^m possible assignments. The partitioning with conditions reduces this number to p mutually exclusive partitions, where p is usually small.

A wide range of conditional tests can be devised. In our implementation, the rule conditions for speech understanding were for instance expressed in terms of regular expressions matches on the user utterance u_u. Generally speaking, a condition is simply defined as a function mapping state variable assignments to a boolean value.

Effects

The rule effects are defined similarly: given a condition holding on a set of input variables, the associated effects define specific *value assignments* for the output variables. The effects can be limited to a single variable or range over several output variables. The effect can also be void, i.e. trigger no change to the distribution over output values.

Each effect is assigned to a scalar value defining its probability or utility, and several alternative stochastic effects can be defined for the same case. If a unique effect is specified, it is then implicitly assumed to hold with probability 1. These values are parameters which can be either hand-coded or estimated from data.

Example

Assume an action selection model structured with probabilistic rules, which operates on a belief state $\mathbf{b}$ containing the last user act a_u as well as a collection of objects perceived in the environment. The response to a polar question such as "is the object red?" can be captured by the following rules:

$$r_1: \textbf{if } (a_u = \text{VerifyColour}(\text{o}, \text{c}) \wedge \text{o.colour} = \text{c}) \textbf{ then } \{Q(a_m = \text{Confirm}) = 5\} \textbf{ else } \{Q(a_m = \text{Confirm}) = -4\}$$

$$r_2: \textbf{if } (a_u = \text{VerifyColour}(\text{o}, \text{c}) \wedge \text{o.colour} \neq \text{c}) \textbf{ then } \{Q(a_m = \text{Disconfirm}) = 5\} \textbf{ else } \{Q(a_m = \text{Disconfirm}) = -4\}$$

$$r_3: \textbf{if } (a_u = \text{VerifyColour}(\text{o}, \text{c})) \textbf{ then } \{Q(a_m = \text{SayDontKnow}) = 1\} \textbf{ else } \{Q(a_m = \text{SayDontKnow}) = -2\}$$

The rule specifies the following behaviour: if there is a reasonable certainty that the object is (resp. is not) of the correct colour, the system should confirm (resp. disconfirm). In case of uncertainty, it should utter "I don't know". The trade-offs between these actions are encoded by the utility parameters.

To illustrate the rules, assume that the dialogue state contains the two independent variables a_u and $\text{o}_1.\text{colour}$, with the respective distributions:

$$P(a_u = \text{VerifyColour}(\text{o}_1, \text{blue})) = 0.8)$$
$$P(a_u = \text{VerifyColour}(\text{o}_1, \text{black})) = 0.2)$$
$$P(\text{o}_1.\text{colour} = \text{blue}) = 0.75)$$
$$P(\text{o}_1.\text{colour} = \text{green}) = 0.25)$$

It is then trivial to calculate that, in this setting, the best action to perform is Confirm, which has a utility $Q = 1.4$, while $Q(\text{SayDontKnow}) = 1$ and $Q(\text{Disconfirm}) = -0.4$.

3.2 Processing workflow

To ease the design of the architecture, the probabilistic rules are grouped into *models*. A model consists of a set of rules and the specification of a "trigger" variable which causes the activation of the model. For instance, the trigger for the speech understanding model $P(a_u|u_u)$ is the user utterance u_u.

The dialogue system is integrated in an event-driven, blackboard architecture (Buckley and Benzmüller, 2007) revolving around the shared belief state $\mathbf{b}$ represented as a Bayesian Network. This belief state is read and written by all the dialogue models. Once a change occurs on a state variable, the algorithm checks whether there are models triggered by this update. Then, for each triggered model, the rules are applied as follows:

1. For every rule r in the model, we create a rule node ϕ_r and include the conditional dependencies with its input variables. If the rule is a probability rule, the rule node will be a chance node describing the distribution of effects given the input assignment. If the rule is a utility rule, the node will be a utility node describing the utility of action variables.

2. The nodes corresponding to the output variables are created (if they do not already exist). For probability rules, these nodes will be chance nodes with a conditional dependence on the rule node ϕ_r. For utility rules, they will be action nodes, with an outward dependence relation to the rule node ϕ_r.

Once no more models can be triggered, the Bayesian Network is modified to replace the updated variables. Finally, if the network contains action variables, the algorithm searches for their optimal action value and selects them. This selection might trigger other inference steps, and the process is repeated until stability is reached. The procedure is described in Algorithms 1 and 2. Figure 3 illustrates the application of four rules on a belief state.

Once the Bayesian network is updated with the new rules, queries can be evaluated using any standard algorithm for exact or approximate inference.

Algorithm 1 : BELIEFUPDATE $(\mathbf{b}, \mathcal{M})$

Require: $\mathbf{b}$: Current belief state
Require: $\mathcal{M}$: Set of rule-based models
loop
  repeat
    for all model $m \in \mathcal{M}$ **do**
      if m is triggered **then**
        for all rule $r \in m$ **do**
          $\mathbf{b} \leftarrow$ ADDRULE$(\mathbf{b}, r)$
        end for
      end if
    end for
  until no model is triggered
  for all node $x' \in \mathbf{b}$ **do**
    Prune x from $\mathbf{b}$
    Relabel x' into x
  end for
  for all node a : action variable **do**
    Find $a^* = \arg\max_{v \in Val(a)} Q(a{=}v|\mathbf{b})$
    Set $a \leftarrow a^*$
  end for
end loop

Algorithm 2 : ADDRULE $(\mathbf{b}, r)$

Require: $\mathbf{b}$: Current belief state
Require: r: Rule to add
$\mathcal{I}_r \leftarrow$ INPUTVARIABLES(r)
Create node $\phi_r \leftarrow$ RULENODE(r)
Add ϕ_r and dependencies $\mathcal{I}_r \rightarrow \phi_r$ to $\mathbf{b}$
if r is a probability rule **then**
  $\mathcal{O}_r \leftarrow$ OUTPUTVARIABLES(r)
  for all output variable $o \in \mathcal{O}_r$ **do**
    Create node o' if not already in $\mathbf{b}$
    Add o' and dependency $\phi_r \rightarrow o'$ to $\mathbf{b}$
  end for
else if r is a utility rule **then**
  $\mathcal{A}_r \leftarrow$ ACTIONVARIABLES(r)
  for all action variable $a \in \mathcal{A}_r$ **do**
    Create node a' if not already in $\mathbf{b}$
    Add a' and dependency $a' \rightarrow \phi_r$ to $\mathbf{b}$
  end for
end if
return $\mathbf{b}$

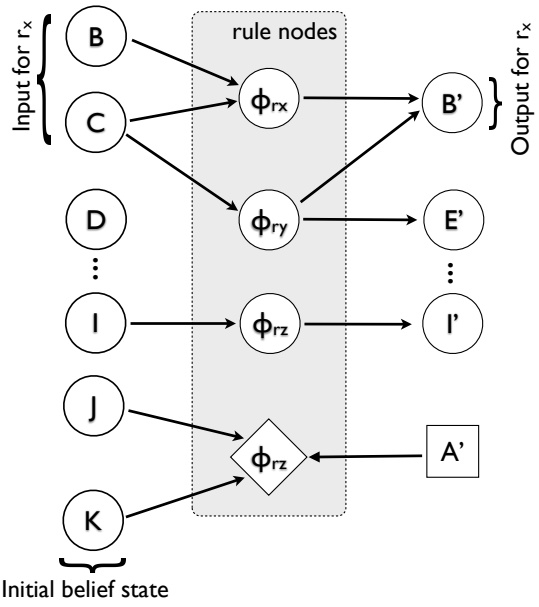

Figure 3: Bayesian Network expanded with the rules r_w, r_x, r_y, r_z on a set of state variables. Diamond nodes represent value nodes, and square action nodes.

It is worth nothing that the outlined procedure is an instance of *ground inference* (Getoor and Taskar, 2007), since the rule structure is grounded in a standard Bayesian Network.

4 Implementation

The framework outlined in the previous section has been implemented in a system architecture, and applied to a human-robot interaction scenario. The scenario involved a human user and a Nao robot[1] (see Figure 4). The user was instructed to teach the robot a sequence of basic movements (lift the left arm, step forward, kneel down, etc.) using spoken commands. The interaction included various dialogue acts such as clarification requests, feedbacks, acknowledgements, corrections, etc.

The models for speech understanding, action selection and generation were all encoded with probabilistic rules, for a total of 68 rules. The expressivity of the formalism allows us to capture complex probability or utility models in just a handful of rules.

[1] A programmable humanoid robot developed by Aldebaran Robotics, `http://www.aldebaran-robotics.com`.

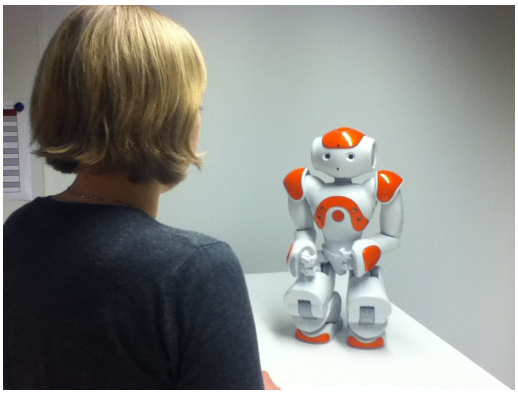

Figure 4: Human user interacting with the Nao robot to teach a sequence of movements via verbal instructions.

The structure of these rules is designed by hand, but their parameters can be learned from data. In our experiments, the utility parameters of the action selection module were estimated from limited amounts of Wizard-of-Oz training data, while the understanding and generation models were hand-crafted – these rules encoding simple, deterministic pattern matching techniques.

In addition, the dialogue also included a speech recognizer (Vocon 3200 from Nuance) connected to the robot microphones, a text-to-speech module, as well as components for planning the robot movements and controlling its motors in real-time. In our experiments, the posteriors for the updated state variables were calculated via importance sampling (Koller and Friedman, 2009), but other inference algorithms such as variable elimination (Zhang and Poole, 1996) are also available.

Both the rule-based models and the external modules are connected to the shared belief state, and read/write to it as they process their data flow. The models and external modules listen for changes occurring in the belief state and become activated when one of these changes relates to their triggering variable. It is worth noting that external modules such as the ASR engine exhibit the same processing behaviour as probabilistic rules, i.e. they extend the belief state with additional rule nodes that themselves lead to updated variables.

4.1 Example of rules

We describe below five examples of rules: two used for shallow dialogue act recognition, two used for action selection, and one for generation.

1. Rule r_1 below lists three regular expression patterns for a particular case of dialogue act classification. If the value for the user utterance variable u_u matches at least one of the patterns, the dialogue act a_u is classified as $\mathrm{LeftArmDown}$, else a_u is left unchanged:

$$\begin{aligned} r_1: \ & \textbf{if } (u_u \text{ matches “left arm down”}) \\ & \vee (u_u \text{ matches “lower * left arm”}) \\ & \vee (u_u \text{ matches “down * left arm”}) \ \textbf{then} \\ & \{P(a'_u{=}\ \mathrm{LeftArmDown}) = 1.0\} \end{aligned}$$

2. Due to high levels of noise in speech recognition, it is often useful to dynamically “prime” the results given expectations from the context. Rule r_2 is another rule for dialogue act recognition triggered if the last system act is $\mathrm{AskRepeat}$. In this case, the rule r_2 will prime the probability that the new user act (a'_u) is identical to the previous one (a_u):

$$\begin{aligned} r_2: \ & \textbf{if } (a_m = \mathrm{AskRepeat}) \ \textbf{then} \\ & \{P(a'_u{=}a_u) = 0.9\} \end{aligned}$$

   Rule r_2 specifies that, if requested to repeat his last dialogue act, the user will do so with probability 0.9. The rule provides a *prediction* of the next user act given the context, before the observation of the user utterance. In combination with classification rules such as r_1, the rule determines the posterior distribution over the most likely dialogue acts uttered by the user.

3. Rule r_3 is an action selection rule which specifies the utility of performing the action $a_m = \mathrm{DoMovement(X)}$ if the user intention i_u is equal to $\mathrm{RequestMovement(X)}$, where X is an argument corresponding to the actual movement (lifting the arm up or down, etc.):

$$\begin{aligned} r_3: \ & \textbf{if } (i_u = \mathrm{RequestMovement(X)}) \ \textbf{then} \\ & \{Q(a'_m{=}\ \mathrm{DoMovement(X)}) = 3.0\} \end{aligned}$$

   Note the use of the unbounded variable X, which is unified at runtime with the actual argument value for i_u.

4. Rule r_4 specifies the utility of the clarification request $a_m = \mathrm{AskRepeat}$. The rule r_4 has no condition, which means that the utility of the clarification request will be conditionally independent of the value of i_u:

$$r_4: \textbf{if } (true) \textbf{ then } \{Q(a'_m = \mathrm{AskRepeat}) = 1.2\}$$

Put together, rules r_3 and r_4 determine the relative utility of requesting a clarification on the user intention vs. performing the action. In this particular case, the system will select $\mathrm{DoMovement(X)}$ if the user intention $\mathrm{RequestMovement(X)}$ has a probability > 0.4, and will ask for a clarification otherwise.

5. Finally, rule r_5 determines the system utterance to synthesise u_m given the system act $a_m = \mathrm{Ack}$ (for "acknowledgement"). In this case, the system is free to select one of the three alternatives, with equal utility:

$$r_5: \textbf{if } (a_m = \mathrm{Ack}) \textbf{ then } \{Q(u'_m = \text{“ok”}) = 1.0 \wedge Q(u'_m = \text{“great”}) = 1.0 \wedge Q(u'_m = \text{“thanks”}) = 1.0\}$$

For the sake of simplicity, the probability and utility values shown above were hand-coded. Of course, dialogue systems deployed in real domains need to estimate these parameters from interaction data (coming from e.g. Wizard-of-Oz experiments). Previous work has demonstrated how to perform such parameter estimation using a Bayesian learning approach (Lison, 2012). One major benefit is that the rule structure is described in exponentially fewer parameters than its plain counterpart, and is thus much easier to learn and to generalise to unseen data.

It should be theoretically possible to also learn the rule *structure* from data, as evidenced by work done in Statistical Relational Learning (Pasula et al., 2007). Such endeavour would however require significantly larger amounts of training data, and remains therefore impractical for most dialogue domains. Furthermore, the rule structure can be seen as a way for the system designer to enforce *design constraints* or business rules into the system (Williams, 2008), and such ability would be lost if the rule structure was to be learned from scratch.

5 Discussion and related work

The development of generic, domain-independent dialogue systems has a long history (Allen et al., 2000; Bohus et al., 2007), and there is a clear trend towards creating platforms with generic or reusable components. There is however no agreement on common modelling formats or processing techniques. In this respect, it is interesting to draw a parallel between dialogue systems and other fields of NLP such as syntactic parsing. Before the 60's, most parsers relied on procedural routines buried in the code. One of the major advances has come from the decision to separate the domain knowledge (in this case, the lexicon and grammar) on one hand, and the parsing algorithms on the other hand. We believe that dialogue systems would also benefit from a cleaner distinction between declarative knowledge (i.e. task- and domain-specific models) and generic processing functionalities (i.e. algorithms for reasoning, learning and planning under uncertainty.)

Most current dialogue systems are however relying on numerous blackbox components in their pipeline. An unfortunate consequence of this heterogeneity is that, while speech recognition results often include explicit measures of uncertainty (in the form of e.g. confidence scores), this uncertainty is often lost at higher stages of reasoning, such as semantic interpretation and dialogue management. Recent papers have shown that confidence scores can be exploited in dialogue management (Williams et al., 2008), but their approach has not yet been widely adopted. Thanks to the unified description framework provided by probabilistic rules, our approach is able to provide a principled account for this uncertainty at all processing stages.

Information-state approaches to dialogue management (Larsson and Traum, 2000; Bos et al., 2003) are closely related to this work, since they also rely on a shared state updated according to a rich repository of rules, but contrary to the approach presented here, these rules are generally deterministic and do not include learnable parameters. The idea of state space partitioning, implemented here via rule conditions, has also been explored in recent

papers (Williams, 2010; Crook and Lemon, 2010).

The work presented in this paper can be seen as an attempt to bridge the gap between "symbolic" approaches to dialogue, which usually concentrate on capturing rich interaction patterns, and "probabilistic" approaches, more focused on aspects related to noise, uncertainty, and learnability. There has been some initial work on hybrid approaches to dialogue processing and management where both statistically learned and designed policies are combined (Williams, 2008; Lee et al., 2010), but they generally use the designed policies as a mere filtering mechanism for the stochastic policy. Our approach however directly incorporates the prior domain knowledge into the statistical model.

Structural knowledge in probabilistic models has been explored in many directions, both in decision-theoretic planning and reinforcement learning (Hauskrecht et al., 1998; Pineau, 2004; Lang and Toussaint, 2010; Otterlo, 2012) and in statistical relational learning (Jaeger, 2001; Richardson and Domingos, 2006; Getoor and Taskar, 2007). The introduced structure may be hierarchical, relational, or both. As in our approach, most of these frameworks rely on expressive representations as *templates* for grounded probabilistic models.

An important side benefit of structured representations in probabilistic models is their improved readability for the human designers, which are able to use these powerful abstractions to encode their prior knowledge of the dialogue domain in the form of pragmatic rules, generic background knowledge, or task-specific constraints. Moreover, the grouping of related rules into models allows the system developer to specify dialogue domains in a modular fashion, by clustering rules into various sets of models. Some models might be highly domain-specific while others encode generic interaction behaviours that can be easily ported to other applications.

6 Conclusion

We have described in this paper a new approach to the development of dialogue systems, based on the declarative specification of *probabilistic rules*. These rules are defined as structured mappings over variables of the dialogue state, specified using high-level conditions and effects. The rules are parametrised with effect probabilities or action utilities. Probabilistic rules allow the system designer to exploit powerful generalisations in the dialogue domain specification without sacrificing the probabilistic nature of the model.

The architecture revolves around a shared belief state expressed as a Bayesian Network. This belief state is continuously updated and extended based on a set of probabilistic rules for speech understanding, management and generation. This architecture has been implemented and integrated in a spoken dialogue system for human-robot interaction. We are currently in the process of refactoring our implementation to make it available as a generic, open source dialogue toolkit called openDial.

We are currently working on extending this architecture in several directions. Our first line of work is to extend the parameter estimation outlined in (Lison, 2012) to Bayesian model-based reinforcement learning. The parameter estimation currently operates in a supervised learning mode, which requires expert data. Alternatively, one could estimate the model parameters in a fully online fashion, without any supervisory input, by incorporating model uncertainty into the inference and continuously adapting the parameter distribution from real or simulated interaction experience (Ross et al., 2011).

Another research direction relates to the extension of the belief update algorithms towards incrementality (Schlangen et al., 2010). We believe that the framework presented in this paper is particularly well suited to perform incremental processing, since the chain of related hypotheses is explicitly captured in the conditional dependencies of the Bayesian Network. A probability change in one initial hypothesis (e.g. the user utterance) will therefore be directly reflected in all hypotheses depending on it (e.g. the corresponding user intention). Extending the belief update algorithm to run incrementally while remaining tractable is however a non-trivial task.

References

J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. 2000. An architecture for a generic dialogue shell. *Natural Language Engineering*, 6:213–228.

D. Bohus, A. Raux, T. K. Harris, M. Eskenazi, and A. I.

Rudnicky. 2007. Olympus: an open-source framework for conversational spoken language interface research. In *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, NAACL-HLT-Dialog '07, pages 32–39, Stroudsburg, PA, USA. Association for Computational Linguistics.

J. Bos, E. Klein, O. Lemon, and T. Oka. 2003. DIPPER: Description and formalisation of an information-state update dialogue system architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, pages 115–124.

M. Buckley and C. Benzmüller. 2007. An agent-based architecture for dialogue systems. In *Proceedings of the 6th international Andrei Ershov memorial conference on Perspectives of systems informatics*, PSI'06, pages 135–147, Berlin, Heidelberg. Springer-Verlag.

P. A. Crook and O. Lemon. 2010. Representing uncertainty about complex user goals in statistical dialogue systems. In *Proceedings of the 11th SIGDIAL meeting on Discourse and Dialogue*, pages 209–212.

L. Getoor and B. Taskar. 2007. *Introduction to Statistical Relational Learning*. The MIT Press.

M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. Dean, and C. Boutilier. 1998. Hierarchical solution of markov decision processes using macro-actions. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 220–229.

M. Jaeger. 2001. Complex probabilistic modeling with recursive relational bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32(1-4):179–220.

D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

G.-J. Kruijff, P. Lison, T. Benjamin, H. Jacobsson, and N. Hawes. 2007. Incremental, multi-level processing for comprehending situated dialogue in human-robot interaction. In *Language and Robots: Proceedings from the Symposium*, Aveiro, Portugal, 12.

T. Lang and M. Toussaint. 2010. Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research*, 39:1–49.

S. Larsson and D. R. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natuarl Language Engineering*, 6(3-4):323–340, September.

C. Lee, S. Jung, K. Kim, and G. Geunbae Lee. 2010. Hybrid approach to robust dialog management using agenda and dialog examples. *Computer Speech & Language*, 24(4):609 – 631.

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

M. Otterlo. 2012. Solving relational and first-order logical markov decision processes: A survey. In *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 253–292. Springer Berlin Heidelberg.

H. M. Pasula, L. S. Zettlemoyer, and L.P. Kaelbling. 2007. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research (JAIR)*, 29(1):309–352, July.

J. Pineau. 2004. *Tractable Planning Under Uncertainty: Exploiting Structure*. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, USA.

M. Richardson and P. Domingos. 2006. Markov logic networks. *Machine Learning*, 62:107–136.

S. Ross, J. Pineau, B. Chaib-draa, and P. Kreitmann. 2011. A Bayesian Approach for Learning and Planning in Partially Observable Markov Decision Processes. *Journal of Machine Learning Research*, 12:1729–1770.

D. Schlangen, T. Baumann, H. Buschmeier, O. Buß, S. Kopp, G. Skantze, and R. Yaghoubzadeh. 2010. Middleware for Incremental Processing in Conversational Agents. In *Proceedings of the 11th SIGDIAL meeting on Discourse and Dialogue*.

V. Thomson and S. Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24:562–588, October.

J. D. Williams and S. Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21:393–422.

J. D. Williams, P. Poupart, and S. Young. 2008. Partially observable markov decision processes with continuous observations for dialogue management. In Laila Dybkjr and Wolfgang Minker, editors, *Recent Trends in Discourse and Dialogue*, volume 39 of *Text, Speech and Language Technology*, pages 191–217. Springer Netherlands.

J. D. Williams. 2008. The best of both worlds: Unifying conventional dialog systems and POMDPs. In *International Conference on Speech and Language Processing (ICSLP 2008)*, Brisbane, Australia.

J. D. Williams. 2010. Incremental partition recombination for efficient tracking of multiple dialog states. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5382–5385.

N. Lianwen Zhang and D. Poole. 1996. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research (JAIR)*, 5:301–328.