UiO : University of Oslo

Towards Dialogue Management in Relational Domains

Pierre Lison,
Language Technology Group (LTG)
Department of Informatics

Workshop on Action, Perception & Language (APL)
October 25 2012

Introduction

- *Slot-filling applications* are still often considered as «prototypical» domains for dialogue systems
- Their representation of the dialogue state is based on particular modelling assumptions:

Task	One single task: filling the predefined slots
Context	None (or very limited): no external context to capture
User model	None (or very limited): different user for each interaction

Introduction

- But these modelling assumptions do not hold for many other domains - especially *situated domains*
- Ex: human-robot interaction, cognitive assistants & companions, tutoring systems, etc.

Task	One single task: filling the predefined slots	*Varying number of interconnected tasks*
Context	None (or very limited): no external context to capture	*Rich, dynamic, often situated environment*
User model	None (or very limited): different user for each interaction	*longer interactions → complex user modelling*

Introduction

- These situated domains have a rich internal structure
- This structure is often best described in terms of *entities* and *relations* between entities:
  - Physical objects spatially connected in a visual scene
  - Indoor environments with places in which to navigate
  - Stacks of (interconnected) tasks to complete

→ Dialogue state expressed as a **relational structure**

The problem

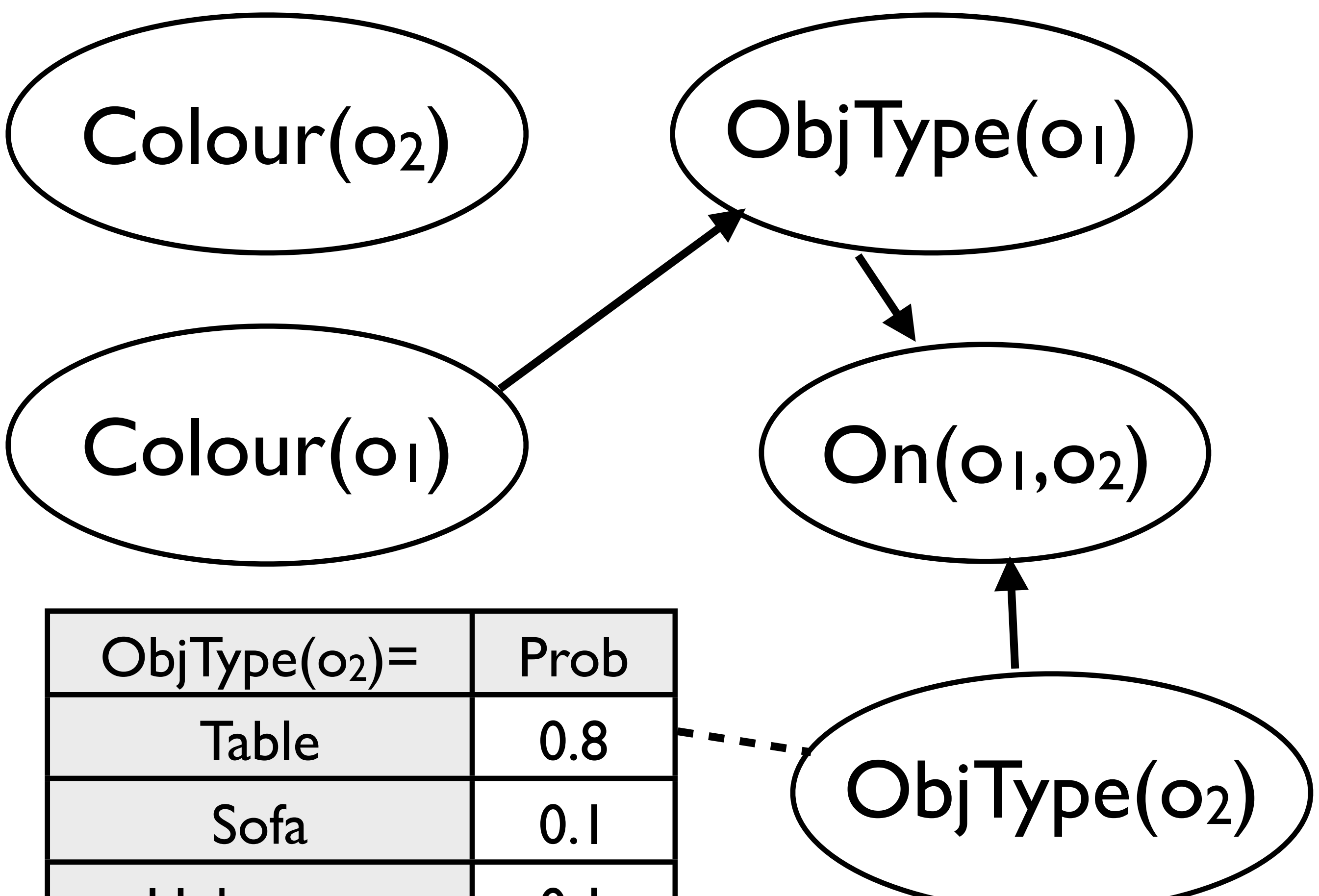
- **Starting point**: we wish to encode the dialogue state as a *relational structure*
- **Problem 1**: how to represent this dialogue state in practice, without giving up the probabilistic modelling?
- **Problem 2**: how do we build *dialogue models* (for interpretation & action selection) that can operate on such state representation?

Problem 1: dialogue state

- **Problem 1**: We need a representation of the dialogue state that is able to:
  - capture the domain's relational structure of the domain
  - account for the pervasive uncertainty in spoken dialogue, especially in situated domains
- We encode our dialogue state as a **Bayesian Network** (i.e. a *directed graphical model*)
- The variables of this network are *grounded predicates and functions*

Problem 1: dialogue state

Assume we want to encode in our dialogue state the type & colour of these two objects, as well as their relative spatial position

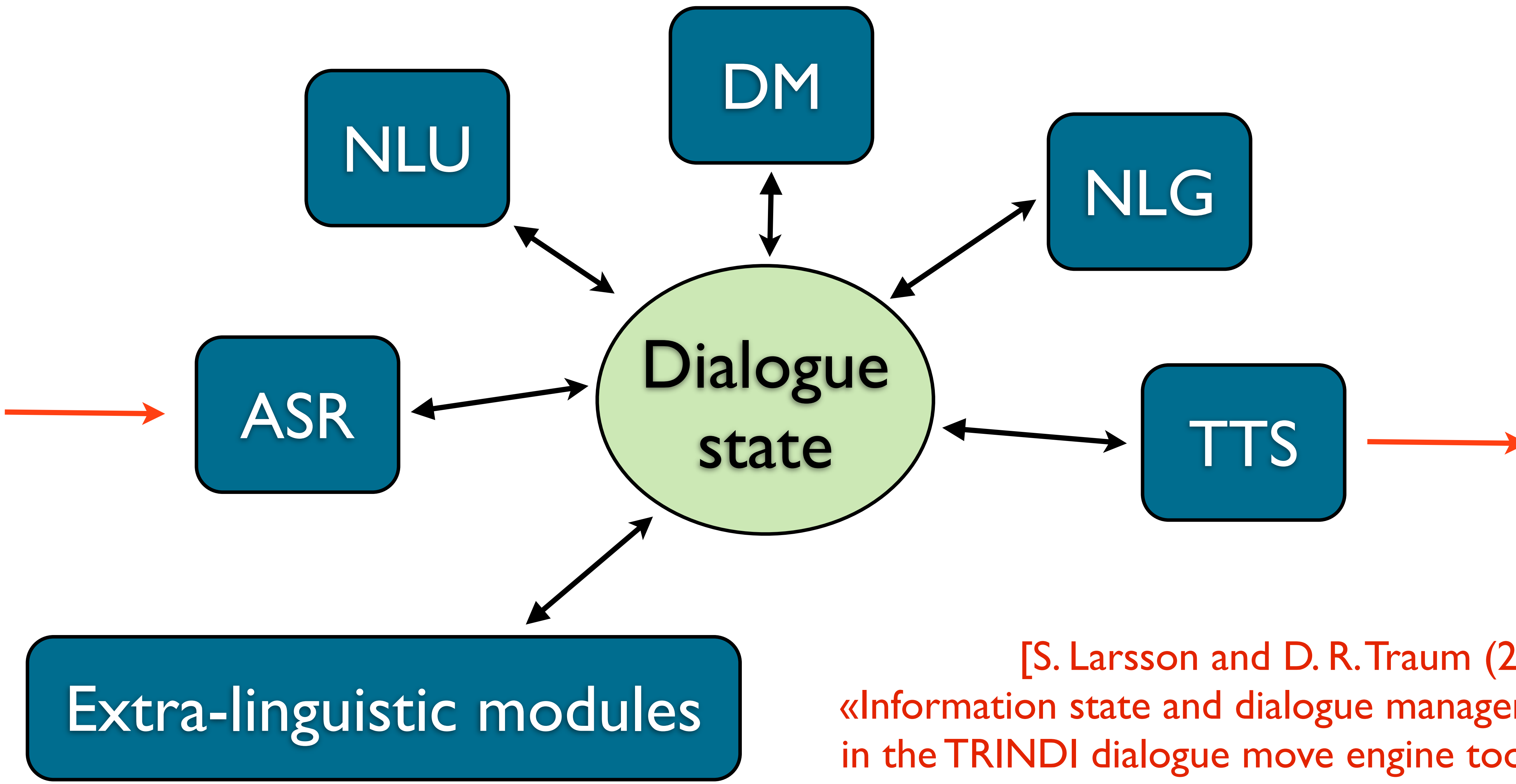


ObjType(o_2)=	Prob
Table	0.8
Sofa	0.1
Unknown	0.1

- The variable labels are ground predicates or functions
- Each variable is associated with a *probability distribution* defining its possible values
- The variable values might depend on each other (*conditional dependencies*)

Problem 1: dialogue state

- *Information-state* architecture: the state acts as a *blackboard* read and written by the various processing components



[S. Larsson and D. R. Traum (2000), «Information state and dialogue management in the TRINDI dialogue move engine toolkit» in *Natural Language Engineering*]

Problem 2: dialogue models

- **Problem 2**: how do we build practical dialogue models defined on the dialogue state representation we just presented?
  - For dialogue interpretation, action selection, etc.
- What we want:
  - *Probabilistic models* of dialogue processing
  - ... that have *parameters* that can be estimated from data
  - ... and take advantage of the domain's *internal structure*

Problem 2: our solution

- **Proposed solution**: encode the dialogue models via expressive *probabilistic* *rules* with a limited form of *quantification*

Approach grounded in *probabilistic modelling*:

- principled account of uncertainties
- parameters can be estimated from data

... but also employing *expressive rules* that can compactly capture

- high-level generalisations
- prior domain knowledge

... and that can range over arbitrary *sets of entities*

Probabilistic rules

- Probabilistic rules take the form of structured **if...then...else** cases
- Mapping from *conditions* to (probabilistic) *effects*:

if (condition$_1$ holds) **then**
  P(effect$_1$)= θ_1, P(effect$_2$)= θ_2, ...
else if (condition$_2$ holds) **then**
  P(effect$_3$) = θ_3, ...
...
else
  P(effect$_n$) = θ_n, ...

Probabilistic rules

- *Conditions* are (arbitrarily complex) logical formulae on state variables
- *Effects* are value assignments on state variables
- Effect probabilities are *parameters* that can be estimated from data

Example:

if (a_m = AskRepeat) **then**
$P(a_u' = a_u) = 0.9$
$P(a_u' \neq a_u) = 0.1$

Utility rules

- The formalism can also describe *utility models*
- In this case, the rule maps each condition to an assignment of *utility values* for particular actions:

if (condition$_1$ holds) **then**
  $Q(\text{actions}_1) = \theta_1$, $Q(\text{actions}_2) = \theta_2$, ...
else if (condition$_2$ holds) **then**
  $Q(\text{actions}_3) = \theta_3$, ...
...
else
  $Q(\text{actions}_n) = \theta_n$, ...

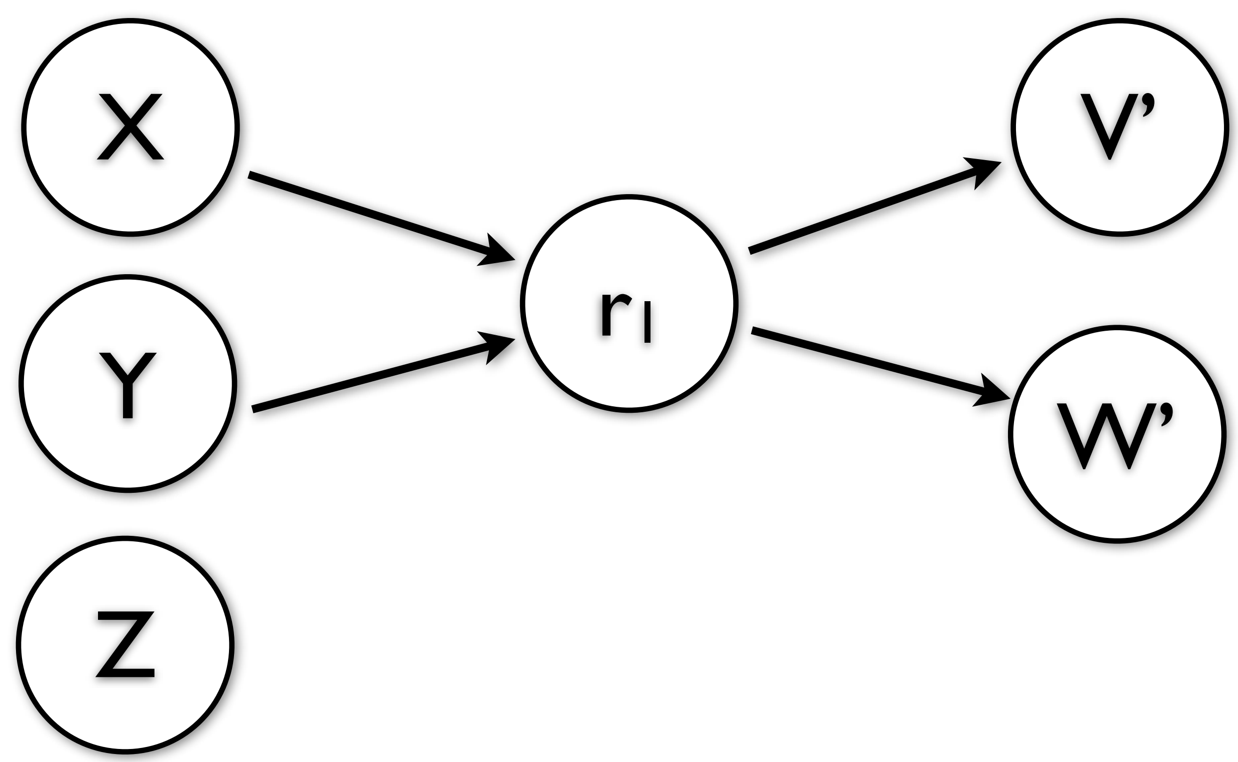
Rule instantiation

- How are the rules applied to the dialogue state?
- The rules are *instantiated* in the Bayesian Network, expanding it with new nodes and dependencies

r_1:

if $(X = \ldots \vee Y \neq \ldots)$ **then**
$P(V = \ldots \wedge W = \ldots) = 0.6$

(The ... dots in r_1 should be replaced by concrete values)



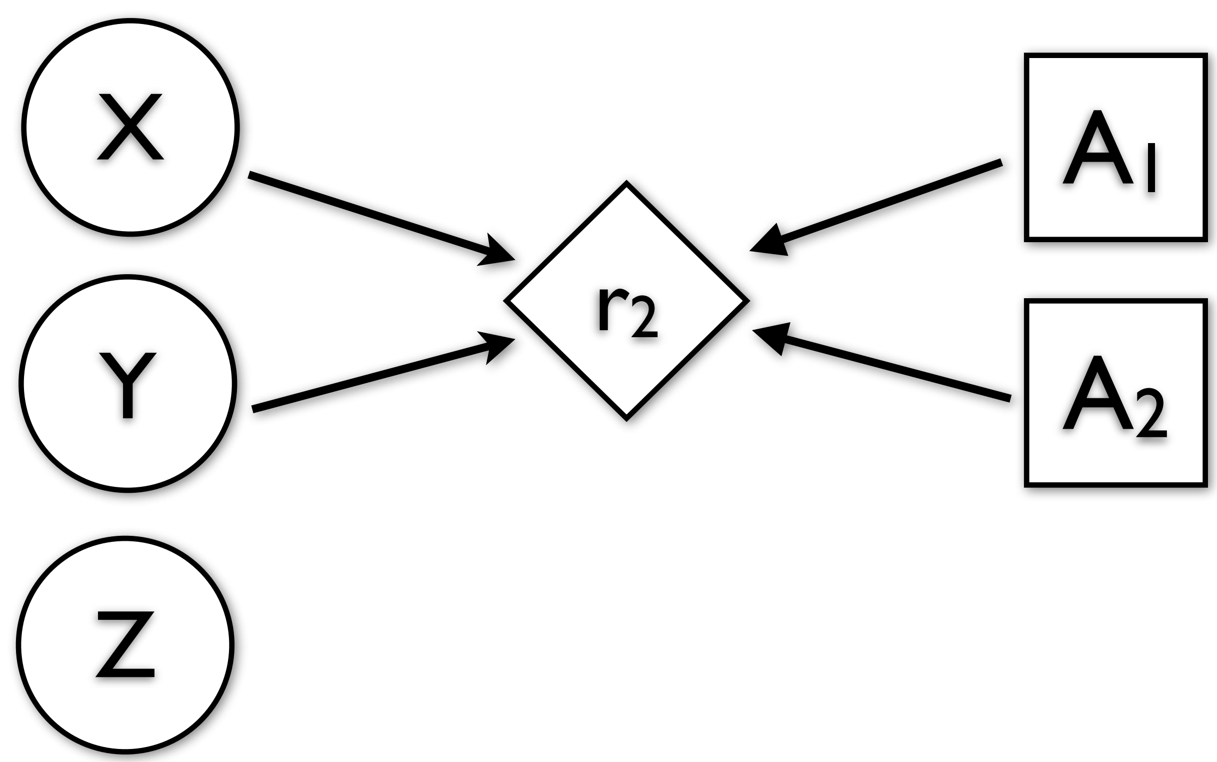
Rule instantiation

- The instantiation procedure is similar for utility rules, although one must employ utility and decision nodes:

r_2:

if $(X = \ldots \vee Y \neq \ldots)$ **then**

$Q(A_1 = \ldots \wedge A_2 = \ldots) = 3$



Quantification mechanism

- If our domain has a relational structure, the rules must be able to abstract over its *entities*
- To this end, we propose to extend probabilistic rules with a limited form of universal quantification:

$\forall\, \mathbf{x} = x_1, \ldots x_k$:

if (condition$_1$($\mathbf{x}$) holds) **then**

P(effect$_1$($\mathbf{x}$))= θ_1, P(effect$_2$($\mathbf{x}$))= θ_2, ...

else if (condition$_2$($\mathbf{x}$) holds) **then**

P(effect$_3$($\mathbf{x}$)) = θ_3, ...

...

else

P(effect$_n$($\mathbf{x}$) = θ_n, ...

Quantification mechanism

- The quantification allows certain variables $x_1, \ldots x_k$ to be *underspecified*.
- The rule will be instantiated for every possible assignment of the underspecified variables.

Example: $\forall o, c :$

$$\textbf{if } (a_m = \text{WhatIsColour}(o) \land o.colour = \text{c}) \textbf{ then}$$
$$\{P(a'_u = \text{Assert}(\text{Is}(o, \text{c}))) = 0.9\}$$

The rule will be instantiated for every possible object *o* and colour *c* matching the condition

Quantification mechanism

- Why is this quantification mechanism useful?
  - Because it allows the system designer to exploit *high-level abstractions* to encode his domain knowledge
  - Because it is a powerful form of *parameter sharing*, which reduces the number of parameters to estimate... and thereby enables learning algorithms to generalise better and with fewer data

Quantification mechanism

- ... but several questions remain to be addressed (work in progress!)
- Main question: how to keep the formalism tractable?
  - If some variables are underspecified, the algorithm must instantiate the rules for every assignment
  - Need to devise *agressive pruning* techniques to quickly discard irrelevant instantiations

Future evaluation

- The presented framework is being implemented in a dialogue system toolkit called **openDial**
- Evaluation in a human-robot interaction scenario

Conclusion & future work

- We have presented here a simple *quantification mechanism* to augment the expressivity of probabilistic rules
- Such mechanism would enable the rules to directly operate on dialogue states represented as *relational structures*
- Ongoing work on implementation and evaluation