# Adaptive Dialogue Strategy Selection through Imprecise Probabilistic Query Answering

O'Neill, I., Yue, A., Liu, W., & Hanna, P. (2011). Adaptive Dialogue Strategy Selection through Imprecise Probabilistic Query Answering. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 11th European Conference, ECSQARU 2011, Belfast, UK, June 29–July 1, 2011. Proceedings* (Vol. 6717, pp. 675-687). Springer. https://doi.org/10.1007/978-3-642-22152-1_57

**Published in:**
Symbolic and Quantitative Approaches to Reasoning with Uncertainty

# Adaptive dialogue strategy selection through imprecise probabilistic query answering

Ian O'Neill[1], Anbu Yue[1], Weiru Liu[1], and Phil Hanna[1]

[1]School of Electronics, Electrical Engineering and Computer Science,
Queen's University Belfast, Belfast BT7 1NN, UK
{i.oneill, a.yue, w.liu, p.hanna}@qub.ac.uk

**Abstract.** In a human-computer dialogue system, the dialogue strategy can range from very restrictive to highly flexible. Each specific dialogue style has its pros and cons and a dialogue system needs to select the most appropriate style for a given user. During the course of interaction, the dialogue style can change based on a user's response and the system observation of the user. This allows a dialogue system to understand a user better and provide a more suitable way of communication. Since measures of the quality of the user's interaction with the system can be incomplete and uncertain, frameworks for reasoning with uncertain and incomplete information can help the system make better decisions when it chooses a dialogue strategy. In this paper, we investigate how to select a dialogue strategy based on aggregating the factors detected during the interaction with the user. For this purpose, we use *probabilistic logic programming (PLP)* to model probabilistic knowledge about how these factors will affect the degree of freedom of a dialogue. When a dialogue system needs to know which strategy is more suitable, an appropriate query can be executed against the PLP and a probabilistic solution with a degree of satisfaction is returned. The degree of satisfaction reveals how much the system can trust the probability attached to the solution.

## 1 Introduction

There are many different ways in which a computer can talk to people. Often dialogue strategies can be categorized as finite-state or frame-based. Additionally, for very fluid, discursive dialogues, a free-form dialogue strategy is appropriate: this may be coupled to techniques for topic recognition as well as mechanisms for transferring to more structured or constrained dialogue once a known transaction context has been identified e.g., [PR03, OHSG].

This paper describes work undertaken by Queen's University Belfast as part of a 3-month collaborative research project commissioned by AUDI AG, Ingolstadt, Germany. While a business-strength solution would entail a functionally richer application and an extensive evaluation programme, the exploratory dialogue system that resulted from this short collaboration served to illustrate how a probabilistic logic program (PLP) might be used to drive a dialogue strategy selection mechanism based on uncertain observations and inputs.

In particular, we were interested in investigating how *probabilistic logic programming* might be able to draw together, into one decision-making process, dialogue-influencing inputs of quite disparate natures and modalities. A dialogue system capable

of replicating a good human listener's sensitivity towards the needs and expectations of a dialogue partner, as well as replicating a human listener's awareness of her/his own limitations, might have to take into account a number of influencing factors. The research programme was not concerned in the first instance with how these factors might be measured or quantified, rather it is on how together these factors influence the selection of a dialogue strategy. The eight core dialogue factors, and the values they could take, are shown below.

**Table 1.** Dialogue-influencing factors

| Factor | Values | Explanations |
|---|---|---|
| Experience (Proficiency) | high (2), med (1), low (0) | How effectively does the user interact with the system? |
| Recognition confidence | high (2), med (1), low (0) | How confident is the system that it has recognised what the user said? |
| Key values | multiple (2), one (1), none (0) | How many usable values does the user provide per turn? |
| Affect | good (2), ok (1), bad (0) | Is the user in a good or bad mood? |
| Response Type | good (3), talkative (2), thinking (1), wondering (0) | What is the relationship between speech and silence in the user's utterances? |
| Productivity $t_0$ | yes (0), no (1) | A turn is productive if a keyword is provided incontext. |
| Productivity $t_{-1}$ | yes (0), no (1) | A turn is productive if a keyword is provided incontext. |
| Productivity $t_{-2}$ | yes (0), no (1) | A turn is productive if a keyword is provided incontext. |

Productivity $t_0$ , Productivity $t_{-1}$ , and Productivity $t_{-2}$ , indicate whether or not usable keywords were identified by the system in the current dialogue turn ($t_0$), and in the two preceding turns ($t_{-1}$ and $t_{-2}$). In the experiment, non-productivity (a failure in dialogue development) was regarded as more significant than productivity (normal dialogue flow). Non-productivity was therefore represented by '1', rather than the default assignment '0'.

For each of the factors indicated above, developers were able to suggest whether a high or low value would be a positive or negative influence on (i.e. should increase or decrease) the *degree of dialogue freedom* in the exchanges between system and user. A freer dialogue would be characterised by system turns that included minimal system prompts, similar to a frame-based dialogue (i.e. just ask the user for the information required, without setting out specific options), while a less-free dialogue would entail a high level of system guidance - to the extent of asking the user for a yes/no response to a very specific question. The role of the PLP was to calculate an overall *degree of dialogue freedom* based on these disparate input factors (which may be uncertain) and their supposed individual influences on the dialogue strategy. In turn the degree of freedom was used to determine the basic dialogue strategy. *Braking factors* were added to the *degree of freedom* calculation, so as to prevent too fast a transition from one dialogue strategy to another. In addition, a *manner* of system delivery (the particular form of words used to realize the dialogue strategy) was influenced by the user's own dialogue manner, perceived affective state, and the frequency with which they used the system. These additional influences on the precise form of the system utterance are

not considered in this paper. Later, however, taking some simplified examples, we will examine the techniques used to calculate the degree of freedom itself.

Since these factors affecting the selection of a dialogue strategy can be modelled using conditional formulae with probabilistic intervals, our research and development is concerned with the appropriateness of using a PLP to help select a dialogue strategy.

Conditional probabilistic logic programming is a framework to represent and reason with imprecise (conditional) probabilistic knowledge. An agent's knowledge is represented by a *probabilistic logic program* (PLP) which is a set of (conditional) logical formulae with probability intervals. The impreciseness of the agent's knowledge is explicitly represented by assigning a probability interval to every logical formula (representing a conditional event) indicating that the probability of a formula will be in the given interval.

To intuitively explain how PLP can be used to model probabilistic knowledge, we take the common knowledge *typically Birds fly, magpies and penguins are birds, but penguins do not fly* as an example to illustrate the meanings of notations. Assume that based on common knowledge, we know that over 98% of birds can fly (so not all birds can fly), and we also know that every *magpie is a bird*. Then this knowledge can be modelled using a PLP as

$$\{(fly(X)|bird(X))[0.98, 1], (bird(X)|magpie(X))[1, 1]\}$$

which can be used to answer queries like *Can a magpie fly?* (e.g., $?(fly(t)|magpie(t))$).

Similarly, within the context of dialogue systems, the relationship between a factor and a dialogue strategy can be modelled using conditional probabilistic logical formulae too. For instance,

$$(dss(t1, free)|exp(t0, high), recog(t0, high))[0.85, 1]$$

states that when both a user's experience and the systems's recognition confidence are high, then the *degree of dialogue freedom* suggests that a *free dialogue strategy* should be chosen with a probability in the interval $[0.85, 1]$.

The main contributions of this paper are as follows. First, we designed a Dialogue Manager for an in-car dialogue system that choose a dialogue strategy dynamically considering factors observed during interaction with the user. With a freer dialogue, the user can provide 'over-informative answers' in response to a single question: however, with a more restricted dialogue, the user needs to answer the question directly. Second, since there is an exponentially large number of combinations of correlated properties in a dialogue, our system allows experts to use PLPs to state probabilistically how each individual property will affect the selection of the dialogue strategy. Third, our PLP *ignorance analysis* tool provides a mechanism that allows experts to judge the quality of the knowledge on which the choice of dialogue strategy is based. Finally, with the assistance of the *degree of satisfaction*, the user can easily configure this system to be freer or more restricted at any time.

This paper is organized as follows. After a brief review of probabilistic logic programming in Section 2, we discuss the role PLPs can play dialogue systems in Section 3. The experiment and simulated evaluations are discussed in Section 4. We compare our work with related research and conclude the paper in Section 5.

## 2 Preliminaries

We briefly review conditional probabilistic logic programming here [Luk98, Luk01, KIL04].

Let $\Phi$ be a finite set of *predicate symbols* and *constant symbols*, and $\mathcal{V}$ be a set of *variables*. An *event* or *logic formula* can be defined from $\Phi \cup \mathcal{V}$ using none or any connectives $\neg, \wedge, \vee$ as usually done in first-order logics. We use $\phi, \psi, \varphi$ for events. For instance, let *Peter* be a person's name, then $man(Peter)$ is a logical formula saying that *Peter is a man* or let $X$ be a variable, then $man(X)$ states that predicate $man$ is applied to variable $X$ . When reasoning, the variable can be bound to a constant. For instance, $man(Peter)$ can be considered as the result of assigning $Peter$ to $X$.

Given a PLP and a query against the PLP, traditionally, either a probability interval or a maximum entropy based probability (denoted as MEP below) is returned as the answer. An interval implies that the true probability of the query shall be within the given interval. However, when this interval is too wide, it provides no useful information. On the other hand, when the knowledge in a PLP is very imprecise, providing a single probability as the solution to a query can be misleading. In [YLH08, YLH10], we developed a new approach which can measure the degree of satisfaction of a single probability solution w.r.t the knowledge provided in a PLP.

A probability distribution $Pr$ satisfies probabilistic formula $(\psi|\phi)[l, u]$ iff $Pr(\psi|\phi) \in [l, u]$. We say that a probabilistic formula $(\psi|\phi)[l, u]$ is a *consequence* of a PLP $P$, denoted as $P \models (\psi|\phi)[l, u]$, iff every probability distribution $Pr$ that satisfies $P$ also satisfies the probabilistic formula. A probabilistic formula $(\psi|\phi)[l, u]$ is a *tight consequence* of $P$, denoted as $P \models_{tight} (\psi|\phi)[l, u]$, iff $P \models (\psi|\phi)[l, u]$ and for all $[l', u'] \subset [l, u]$, $P \not\models (\psi|\phi)[l, u]$. For simplicity, if $P \models (\phi|\top)[0, 0]$, we denote $P \models_{tight} (\psi|\phi)[1, 0]$.

We use $me[P]$ to denote the probability distribution with maximum entropy among those that satisfy $P$. Let $P$ be a PLP, we say that $(\psi|\phi)[l, u]$ is a *me-consequence* of $P$, denoted as $P \models^{me} (\psi|\phi)[l, u]$, iff $P$ is unsatisfiable or $me[P] \models (\psi|\phi)[l, u]$. We say that $(\psi|\phi)[l, u]$ is a *tight me-consequence* of $P$, denoted as $P \models^{me}_{tight} (\psi|\phi)[l, u]$, iff one of the following conditions holds:

- $P \models (\phi|\top)[0, 0]$, $l = 1$, $u = 0$,
- $me[P](\phi) > 0$ and $me[P](\psi|\phi) = l = u$.

*Example 1.* Let PLP $P$ be defined as follows:

$$P = \left\{ \begin{array}{l} (fly(X)|bird(X))[0.98, 1] \\ (bird(X)|penguin(X))[1, 1] \\ (penguin(X)|bird(X))[0.1, 1] \end{array} \right\}$$

Based on this knowledge base, a user can query the likelihood that *a penguin can fly*, e.g, $?(fly(t)|penguin(t))$.

The results of using our prediction tool based on this knowledge base is $P \models_{tight} (fly(t)|penguin(t))[0, 1]$, and $P \models^{me}_{tight} (fly(t)|penguin(t))[0.98, 0.98]$.

In [KIL04, Luk98, Luk01], approaches were provided to calculate the probability interval and probability with maximum entropy for any query. In [YLH08, YLH10], a formal method was provided to analyze the PLP and the maximum entropy principle as

well as to calculate the degree of ignorance and degree of sastisfaction reviewed in this section.

First, an ignorance value is provided to evaluate the extent to which the answer given under maximum entropy is reliable. Second, a measure of the degree of satisfaction is provided to evaluate how reliable an interval is to serve as the answer of the query.

**Definition 1 (Ignorance).** *Let $\mathcal{PL}$ be the set of all PLPs and $\mathcal{E}$ be a set of conditional events. Function $\mathsf{IG} : \mathcal{PL} \times \mathcal{E} \mapsto [0, 1]$ is called the* measure [1]of ignorance*, iff for any PLP $P$ and conditional event $(\psi|\phi)$ it satisfies the following postulates*

**[Bounded]** $\mathsf{IG}(P, \psi|\phi) \in [0, 1]$.
**[Preciseness]** $\mathsf{IG}(P, \psi|\phi) = 0$ *iff* $P \models_{tight} (\psi|\phi)[u, u]$ *or* $P \models \phi \rightarrow \bot$.
**[Totally Ignorance]** $\mathsf{IG}(\emptyset, \psi|\phi) = 1$, *if* $\not\models \phi \rightarrow \psi$ *and* $\not\models \phi \rightarrow \neg\psi$.
**[Sound]** *If* $\mathsf{IG}(P, \psi|\phi) = 1$ *then* $P \models (\psi|\phi)[0, 1]$.
**[Irrelevance]** *If $P$ and another PLP $P'$ do not contain common syntaxes,*
*i.e.*$\Phi \cap \Phi' = \emptyset$*, then* $\mathsf{IG}(P, \psi|\phi) = \mathsf{IG}(P \cup P', \psi|\phi)$.

If $P = \emptyset$, only tautologies can be inferred from $P$. Therefore, from any PLP $P$, $\mathsf{IG}_P(\psi|\phi) \leq \mathsf{IG}_\emptyset(\psi|\phi)$, which means that an empty PLP has the biggest ignorance value for any conditional event. When $\mathsf{IG}_P(\psi|\phi) = 0$, event $(\psi|\phi)$ can be inferred precisely from $P$, since a single precise probability for $(\psi|\phi)$ can be obtained from $p$. The ignorance measurement focuses on the knowledge about $(\psi|\phi)$ contained in $P$, which means that irrelevant knowledge does not provide a better understanding of this conditional event.

**Definition 2 (Degree of Satisfaction).** *Let $\mathcal{PL}$ be the set of all PLPs and $\mathcal{F}$ be a set of probabilistic formulae. Function $\mathsf{SAT} : \mathcal{PL} \times \mathcal{F} \mapsto [0, 1]$ is called the measure of* degree of satisfaction *iff for any PLP $P$ and ground probabilistic formula $\mu = (\psi|\phi)[l, u]$, it satisfies the following postulates:*

**[Reflexive]** $\mathsf{SAT}(P, \mu) = 1$, *iff* $P \models \mu$.
**[Rational]** $\mathsf{SAT}(P, \mu) = 0$ *if* $P \cup \{\mu\}$ *is unsatisfiable.*
**[Monotonicity]**
$\mathsf{SAT}(P, \mu) \geq \mathsf{SAT}(P, (\psi|\phi)[l', u'])$, *if* $[l', u'] \subseteq [l, u]$.
$\mathsf{SAT}(P, \mu) > \mathsf{SAT}(P, (\psi|\phi)[l', u'])$, *if* $[l', u'] \subset [l, u]$
*and* $\mathsf{SAT}(P, (\psi|\phi)[l', u']) < 1$.
**[Cautious Monotonicity]** *Let* $P' = P \cup \{(\psi|\phi)[l', u']\}$*, where* $P \models^{me} (\psi|\phi)[l', u']$
*If* $1 \geq \mathsf{SAT}(P, \mu) \geq 0$ *then* $\mathsf{SAT}(P', \mu) \geq \mathsf{SAT}(P, \mu)$,

The reflexive property says that every consequence is totally satisfied. The rational property says that 0 is given as the degree of satisfaction of an unsatisfiable probabilistic formula.

Monotonicity says that if we expect a more precise interval for a query, then the chance that the exact probability of the query is *not* in the interval is getting bigger.

[1] In mathematical analysis, a measure $m$ is a function, such that $m : 2^S \mapsto [0, \infty]$ and
1. $m(E_1) \geq 0$ for any $E \subseteq S$;
2. $m(\emptyset) = 0$;
3. If $E_1, E_2, \ldots$ is a countable sequence of pairwise disjoint subsets of $S$, the measure of the union of $E_i$'s is equal to the sum of the measures of each $E_i$, that is, $m(\bigcup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} m(E_i)$

Cautious monotonicity says that, if $P$ and $P'$ are equivalent except for the bound of $(\psi|\phi)$, and if $P'$ contains more knowledge about $(\psi|\phi)$, then the degree of satisfaction of $\mu$ under $P'$ should be bigger than that of $\mu$ under $P$.

*Example 2.* Let $P$ and query $?Q$ be the same as in Example 1. Then, the degree of satisfaction for the query answer $[0.7, 1]$ is $0.8$.

If we require that the degree of satisfaction of a query answer must be above a threshold $\gamma$, then the PLP reasoning system can produce a tightest interval for which its degree of satisfaction is not less than $\gamma$. When $\gamma = 0.5$, the returned interval is [MEP, 1] (the upper bound is set to 1 in our system), so we obtain both an interval and the MEP value. This kind of consequence relation is an extension to $\models_{tight}$. Details about the calculation of degree of satisfaction are available in [YLH08, YLH10].

## 3 Dialogue systems

A fully implemented spoken dialogue system requires a delicately balanced interaction between a number of main components. These typically include a speech recogniser, a semantic parser, a dialogue manager, a natural language generator, a speech synthesiser and an underlying database. More recently, components intended to capture and synthesise non-verbal interaction - affect recognisers, embodied conversational agents, and so on - may also feature in the configuration. In the experiment described here, we focus on the behaviour manifested by the Dialogue Manager, which takes a co-ordinating and decision-making role, determining how the system should ask the user questions and respond to the user's answers.

We were particularly interested in the degree of freedom that the Dialogue Manager should offer the user as they conducted their conversation. Thus the Dialogue Manager had at its disposal a number of dialogue strategies, ranging from several flavours of tightly system-led 'finite state' approaches, which required that the user choose just one of the options presented on a particular dialogue turn, to freer 'frame-based 'solutions, where, without being explicitly told the available options, the user could supply the system with one or more values needed to populate 'slots' in a notional enquiry frame.

### 3.1 Using probabilistic logic programs to represent imprecise data

The permutations of all dialogue-influencing factors are exponentially large in number and far exceed the experts' power to define a strategy for each of them. Developers can however provide PLP representations of the typical effect on dialogue style of key individual dialogue-influencing factors and of key combinations of these factors. Using this information the system can calculate degrees of dialogue freedom for all combinations of dialogue-influencing factors.

One problem in using traditional probabilistic logic programming [CPQC03,Luk01] is that, only a loose and uninformative interval or an unreliable single probability value can be extracted as the answer. However, by using PLP to model the domain knowledge and by then applying our reasoning method, we obtain more reliable intervals and single values in response to a query.

### 3.2 Observation vs. a priori facts

In PLPs, we use ground formulae to state a priori facts from statistics, i.e., something that must be true (statistically) is regarded as a fact. These facts are treated differently from observations about individuals. Observing an event (such as the total number of

recognized keywords by a user) does not infer that the event would happen for sure. So, observations cannot be represented as formulae of the form $(\psi(a)|\top)[1,1]$ in a PLP: doing so implies that we know $\psi(a)$ to be true even before it is observed. In other words, taking $\psi(a)$ as a probabilistic event, we cannot predict if $\psi(a)$ is true or false before we observe it. In dialogue systems, observations are very important for choosing dialogue strategies. In our framework, all observations are stored in a database (named $OBS$) that is separate from a PLP containing statistical knowledge. When querying $(\psi|\phi)[l,u]$ on PLP $P$, this observation database $OBS$ is automatically called, with the effect that querying $(\psi|\phi)[l,u]$ is equivalent to querying $(\psi|\phi \wedge \bigwedge OBS)[l,u]$ on $P$.

# 4 The Experiment and Evaluation of our Framework

## 4.1 Conducting the experiment and constructing a PLP

For each possible permutation of the values of the eight factors, the Dialogue Manager would use the answer to the query as its degree of dialogue freedom and would select its dialogue strategy accordingly. We interpret the answer to a query as: *With a given degree of satisfaction, what is the best estimation of the probability that a free dialogue strategy would be appropriate in these circumstances?* With a $0.5$ degree of satisfaction, the result equals the Maximum Entropy Probability.

However, rather than have the Dialogue Manager generate degrees of dialogue freedom live (a computationally very intensive process), it acquired the answers for the queries from a look-up table, generated beforehand by the PLP Reasoner and Analyzer [YLH08, YLH10]. and covering all possible permutations of the values of the eight dialogue-influencing factors.

Thus, values generated off-line by our Reasoning Engine would subsequently be used live by the Dialogue Manager, as it selected its dialogue strategy. In the following example we concentrate on just a handful of dialogue-influencing factors, the PLP used by the Reasoning Engine, and the output generated by the Reasoning Engine. In the PLP, predicates *exp, recog*, and *key* are used to state respectively the user's experience (strictly speaking, the user's *proficiency*: elsewhere we have used the term *experience* in a simpler sense to represent the number of times the user has interacted with the system); the system's recognition confidence; and the number of keywords recognized from the user's response. The sample probabilistic formulae below reflect these conventions. These formulae represent the effect of *combinations* of dialogue-influencing factors, and similar formulae are used to represent the typical effect on dialogue freedom of *individual* dialogue-influencing factors:

**Table 2.** Probability intervals for a free dialogue strategy given a selection of dialogue-influencing factors.

| | | Factor: User Experience | | | |
|---|---|---|---|---|---|
| | Level | High | Medium | Low | |
| Factor: | High | [0.85, 1] | [0.80, 0.95] | [0.75, 0.85] | |
| Recognition | Medium | [0.65, 0.85] | [0.6, 0.8] | [0.55, 0.75] | Probability |
| Confidence | Low | [0.45, 0.65] | [0.4, 0.6] | [0.35, 0.55] | |
| Factor: | Multiple | [0.95, 1] | [0.90, 0.95] | [0.80, 0.95] | Interval |
| Number of | Single | [0.90, 1] | [0.85, 1] | [0.60, 0.90] | |
| Recognized Keywords | None | [0.80, 1] | [0.70, 0.90] | [0.20, 0.70] | |

If the recognition confidence is high and user experience is high too, then the probability that a free dialogue strategy is appropriate is in the interval [0.8,1]. A PLP capturing this probabilistic knowledge can be created as follows.

$$
\begin{array}{ll}
(dss(t1, free)|exp(t0, high), recog(t0, high)) & [0.85, 1] \\
(dss(t1, free)|exp(t0, high), recog(t0, med)) & [0.65, 0.85] \\
(dss(t1, free)|exp(t0, high), recog(t0, low)) & [0.45, 0.65] \\
(dss(t1, free)|exp(t0, med), recog(t0, high)) & [0.80, 0.95] \\
(dss(t1, free)|exp(t0, med), recog(t0, med)) & [0.60, 0.80] \\
(dss(t1, free)|exp(t0, med), recog(t0, low)) & [0.40, 0.60] \\
(dss(t1, free)|exp(t0, low), recog(t0, high)) & [0.75, 0.85] \\
(dss(t1, free)|exp(t0, low), recog(t0, med)) & [0.55, 0.75] \\
(dss(t1, free)|exp(t0, low), recog(t0, low)) & [0.35, 0.55] \\
(dss(t1, free)|exp(t0, high), key(t0, multiple)) & [0.95, 1.00] \\
(dss(t1, free)|exp(t0, high), key(t0, single)) & [0.90, 1.00] \\
(dss(t1, free)|exp(t0, high), key(t0, none)) & [0.80, 1.00] \\
(dss(t1, free)|exp(t0, med), key(t0, multiple)) & [0.90, 0.95] \\
(dss(t1, free)|exp(t0, med), key(t0, single)) & [0.85, 1.00] \\
(dss(t1, free)|exp(t0, med), key(t0, none)) & [0.70, 0.90] \\
(dss(t1, free)|exp(t0, low), key(t0, multiple)) & [0.80, 0.95] \\
(dss(t1, free)|exp(t0, low), key(t0, single)) & [0.60, 0.90] \\
(dss(t1, free)|exp(t0, low), key(t0, none)) & [0.20, 0.70]
\end{array}
$$

The probability of $dss(t1, free)$ stands for the degree of freedom of the dialogue strategy at the next time point $t1$, while, for our initial trial, the values [min,max] represent the range within which developers believe dialogue freedom should lie, given the current level (high, medium, low, etc.) of the dialogue-influencing factor under consideration in the formula.

In order to facilitate reasoning, we need to include some background knowledge in this PLP for example, that the system's recognition confidence cannot be both high and medium simultaneously. This background knowledge is represented as the following additional probabilistic formulae:

$$
\begin{array}{l}
recog(t0, high), recog(t0, med)[0, 0] \\
recog(t0, low), recog(t0, med)[0, 0] \\
recog(t0, high), recog(t0, low)[0, 0] \\
exp(t0, high), exp(t0, med)[0, 0] \\
exp(t0, low), exp(t0, med)[0, 0] \\
exp(t0, high), exp(t0, low)[0, 0] \\
key(t0, multiple), key(t0, single)[0, 0] \\
key(t0, multiple), key(t0, none)[0, 0] \\
key(t0, single), key(t0, none)[0, 0]
\end{array}
$$

Now assume that we have a user, $A$, whose experience is *medium* at time point $t0$, assume also that keywords recognition is *multiple*, and recognition confidence is *low*. To determine which dialogue strategy is most suitable, we query

$$Q = ?(dss(t1, free)|exp(t0, med), key(t0, multiple), recog(t0, low))$$

This can be executed against the PLP constructed above. For simplicity, let $E$ be the conditional event in this query (i.e., $Q = ?E$). For this query, we find $P \models_{tight} E[0, 1]$ and $P \models^{me}_{tight} E[0.8667, 0.8667]$. That is, we get a non-informative interval $[0, 1]$ and a precise probability $0.8667$ as two possible answers to this query.

Note that the statistical (or, in the test system, heuristic) knowledge in the PLP states that for a user of medium experience, if the system's recognition confidence is low, then the next round of dialogue freedom should be relatively low ($[0.40, 0.60]$). However, if multiple keywords are recognized by the system then the freedom of the next round should be relatively high ($[0.80, 0.95]$). These two rules state two possible degrees of freedom for the next round, but how these two factors should be integrated, in order to determine the degree of freedom of next turn, remains unclear. The value $0.8667$ given by the maximum entropy principle, suggests a degree of compromise.

Now, we examine the degree of satisfaction and ignorance of $?E[l, u]$. When $l = u = 0.8667$, the ignorance of this query is bigger than $0$. Thus the value $0.8667$ is possibly not a very accurate degree of dialogue freedom. To find a probabilistic interval within which the true probability might lie and to quantify our satisfaction with this interval, we assign different values to $l$ and $u$ and calculate the degree of satisfaction and ignorance for each pair $l$ and $u$. Details of the calculation are given in Table 3. From this table, the system can choose a dialogue strategy based on these degrees of satisfaction. For instance, we have $p(Pr(E) \in [0.4, 1] = 0.8034$, which means the probability (that a degree of dialogue freedom of $0.4$ in the next dialogue turn will be appropriate to this user) is $0.8034$. From another perspective, it may be the case that there is a very high probability that the system has clearly recognised the user's response. If this is so, a strategy with freedom $0.4$ may be too restrictive. Therefore, a dialogue system should choose a strategy that balances both the degree of satisfaction and the level of freedom that it affords the user. For a user who prefers a less restricted dialogue, the threshold of the degree of satisfaction can be low and for a user who prefers a system-led dialogue, the threshold can be higher.

**Table 3.** Probability interval for $?E$ and the degree of satisfaction of $Pr(E) \in [l, u]$

| Probability interval | Degree of satisfaction | Probability interval | Degree of satisfaction |
|---|---|---|---|
| $[0.1, 1]$ | 0.9586 | $[0.2, 1]$ | 0.9138 |
| $[0.3, 1]$ | 0.8621 | $[0.4, 1]$ | 0.8034 |
| $[0.5, 1]$ | 0.7310 | $[0.6, 1]$ | 0.6552 |
| $[0.7, 1]$ | 0.5759 | $[0.8, 1]$ | 0.5142 |
| $[0.81, 1]$ | 0.5104 | $[0.82, 1]$ | 0.5072 |
| $[0.83, 1]$ | 0.5045 | $[0.84, 1]$ | 0.5024 |
| $[0.85, 1]$ | 0.5010 | $[0.86, 1]$ | 0.5002 |

### 4.2 Evaluating the results

However, in our initial experiment we accepted the figure for MEP *at face value* and used it to help us select our dialogue strategy. We were particularly interested in the manner in which the precise probability, the MEP, changed according to the dialogue influencing factors that were input. Since we (and the system) take this precise value as answering the question *How probable is it that a free dialogue strategy would be appropriate in these circumstances?*, then, if the value turned out to be 1 (entirely probable),

the Dialogue Manager would, if unchecked by any braking factor, chose the freest strategy; if the value were 0 (entirely improbable), it would choose the most restrictive strategy; and if somewhere between 0 and 1, it would choose a restrictive or non-restrictive strategy corresponding to the band within which the freedom value fell. In reality this *raw* freedom value was generally modified by a braking factor, which varied according to user proficiency, in order, as we have mentioned, to avoid jarringly rapid shifts between quite different dialogue strategies: thus a *raw* freedom value was converted by means of the appropriate braking factor, to a *current* freedom value and it was this that was used to select the dialogue strategy for the next turn. In future the braking factor might be replaced by an additional dialogue-influencing factor to be considered by the PLP Reasoner and Analyzer.

To judge the correctness of the *raw* system behaviour, we monitored the manner in which the unmodified MEP rose or fell depending on the combinations of dialogue influencing factors that were input. In the PLP for the experiment, some individual factors and some combinations of factors had been identified as placing raw *dialogue freedom* within particular ranges (probability intervals). In this way the PLP represented the developers', and by extension the Dialogue Manager's, basic decision-making heuristics or rules of thumb.

Table 4 illustrates the system's raw behaviour in one typical sequence of eight dialogue moves. Input to the system is shown for each move in the form of an eight-element feature vector. In each feature vector the input elements are ordered and have values as described previously in Table 1. Each feature vector is followed by the corresponding raw output MEP.

Table 4 represents eight moves in a dialogue with a proficient user. In user turns U1, U2 and U3 the dialogue is progressing well (raw freedom value 1). In user turns U4 and U5 there is no dialogue product (i.e. no usable user input) and the user's affective state worsens (the freedom value falls). In user turn U6 there is product and the user's affective state improves (the freedom value rises). In user turns U7 and U8 the user's affective state worsens again, and other dialogue factors are also sub-optimal (the freedom value falls again).

**Table 4.** The effect on MEP of different levels of dialogue-influencing factor across eight user-turns

| | Dialogue-Influencing Factors | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| User Turn | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 0.5 (MEP) |
| U1 | 2 | 2 | 2 | 2 | 3 | 0 | 1 | 1 | 1.00000 |
| U2 | 2 | 2 | 2 | 2 | 3 | 0 | 0 | 1 | 1.00000 |
| U3 | 2 | 2 | 2 | 2 | 3 | 0 | 0 | 0 | 1.00000 |
| U4 | 2 | 0 | 0 | 2 | 3 | 1 | 0 | 0 | 0.82560 |
| U5 | 2 | 0 | 0 | 1 | 3 | 1 | 1 | 0 | 0.62300 |
| U6 | 2 | 1 | 1 | 2 | 3 | 0 | 1 | 1 | 0.83770 |
| U7 | 2 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0.70000 |
| U8 | 2 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0.64080 |
| | Level | | | | | | | | |

This scenario and others like it indicate that the degree of raw dialogue freedom, as represented by the generated MEP, is indeed a reasonable interpretation of the developers' rules of thumb for determining when freer dialogue strategies should be allowed, and when more restrictive system-led dialogue strategies should be considered.

## 5 Related Work and Conclusion

**Related work:** Logic programming is now a well established knowledge representation and reasoning formalism in artificial intelligence and deductive databases. The need for representing uncertainty in the logic programming framework is already reported by a great number of publications [Luk98, BGR04, BH07, RKT07, Fuh00], [DD04] etc.

Our PLP analytical and reasoning system [YLH08] is based on *conditional probabilistic logic programming* [CPQC03, Luk01], in which knowledge is represented by interval restrictions for probabilities on conditional events in the form of $(\psi|\phi)[l, u]$. In traditional probabilistic logic programming, the answer for a query is either a very uninformative wide interval or an unreliable probability value. In contrast, our method can provide an ignorance degree to evaluate how useful a PLP is to answer a query, and furthermore provide a reasoning method to give a more informative (narrower) interval as the answer that is acceptable to a user.

A few IT systems have been implemented that model and query probabilistic knowledge, for example, SPIRIT [RRK06] and PIT [SF97].

In order to manage imprecise probabilistic reasoning, an expert system shell, SPIRIT, was implemented which uses the principle of maximum entropy to avoid the request of precise probability distributions. Knowledge acquisition is performed by specifying probabilistic facts and rules on discrete variables in an extended propositional logic syntax. The shell generates the unique probability distribution which respects all facts and rules and maximizes entropy. After creating this distribution the shell is ready for answering simple and complex queries. System PIT (Probability Induction Tool) was implemented based on propositional logic, the probability calculus and the concept of model-quantification. The task of PIT is to deliver decisions under incomplete knowledge but to keep the necessary additional assumptions as minimal as possible.

In contrast, our system deploys a reasoning mechanism in conditional probabilistic logic programming, which is based on first order logic, rather than propositional logic.

From a dialogue perspective, the need to give a dialogue system the correct amount of user-led and system-led interaction is important. An experienced user will become frustrated, if in ideal conditions he or she is unduly restricted by the system, but so will an inexperienced user, if, in adverse conditions, the system does not curtail dialogue freedom (the paths along which the dialogue might progress) by providing assistance and guiding the dialogue more closely to a successful conclusion.

**Conclusion:** For the authors this has been a useful first exploration of the issues involved in PLP-based natural language dialogue management. Some advantages, and potential challenges, for live system development have emerged.

From a dialogue modelling perspective, the decision-making engine, the PLP, can be regarded as a black box. In other words, for those concerned with implementing a naturalistic dialogue, one which can at least pass as a reasonable sequence of spoken exchanges between system and user in pursuance of some task, it is immaterial how the figure representing the level of dialogue freedom is derived, so long as it might

reasonably be regarded (by an external observer) as a sensible compromise between factors that individually pull towards greater dialogue restriction or push towards greater dialogue flexibility.

However in the longer term, logic programmers and dialogue modelers will have to embark on a demanding dialogue of their own, as they attempt to understand each others capabilities as knowledge engineers and meet each others requirements as providers of usable software systems.

Besides furthering this inter-disciplinary dialogue, future work will entail using more realistic, live inputs to represent the various factors that influence the human-computer dialogue. It will also involve attempting to assess how satisfied different categories of actual user are with the variety of dialogue styles that the system adopts in varied and evolving circumstances.

## References


[BGR04] Chitta Baral, Michael Gelfond, and J. Nelson Rushton. Probabilistic reasoning with answer sets. In *LPNMR*, pages 21–33, 2004.

[BH07] Chitta Baral and Matt Hunsaker. Using the probabilistic logic programming language p-log for causal and counterfactual reasoning and non-naive conditioning. In *IJCAI*, pages 243–249, 2007.

[CPQC03] Vítor Santos Costa, David Page, Maleeha Qazi, and James Cussens. CLP($\mathcal{BN}$): Constraint logic programming for probabilistic knowledge. In *UAI*, pages 517–524, 2003.

[DD04] Alex Dekhtyar and Michael I. Dekhtyar. Possible worlds semantics for probabilistic logic programs. In *ICLP*, pages 137–148, 2004.

[Fuh00] Norbert Fuhr. Probabilistic datalog: Implementing logical information retrieval for advanced applications. *JASIS*, 51(2):95–110, 2000.

[KIL04] Gabriele Kern-Isberner and Thomas Lukasiewicz. Combining probabilistic logic programming with the power of maximum entropy. *Artificial Intelligence*, 157(1-2):139–202, 2004.

[Luk98] Thomas Lukasiewicz. Probabilistic logic programming. In *ECAI*, pages 388–392, 1998.

[Luk01] Thomas Lukasiewicz. Probabilistic logic programming with conditional constraints. *ACM Trans. Comput. Log.*, 2(3):289–339, 2001.

[OHSG] I.M. O'Neill, P.J. Hanna, D.W. Stewart, and X. Gu. Use of the quads architecture for multimodal output generation.

[PR03] M. Pantic and L.J.M. Rothkrantz. M. pantic and l.j.m. rothkrantz. In *IEEE*, pages 1370–1390, 2003.

[RKT07] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, pages 2462–2467, 2007.

[RRK06] Wilhelm Rödder, Elmar Reucher, and Friedhelm Kulmann. Features of the expert-system-shell spirit. *Logic Journal of the IGPL*, 14(3):483–500, 2006.

[SF97] Manfred Schramm and Volker Fischer. Probabilistic reasoning with maximum entropy - the system pit (system description). In *WLP*, 1997.

[YLH08] Anbu Yue, Weiru Liu, and Anthony Hunter. Measuring the ignorance and degree of satisfaction for answering queries in imprecise probabilistic logic programs. In *SUM*, pages 386–400, 2008.

[YLH10] Anbu Yue, Weiru Liu, and Anthony Hunter. Imprecise probabilistic query answering using measures of ignorance and degree of satisfaction. *Annals of Mathematics and Artificial Intelligence*, accepted, 2010.