

# An Answer Explanation Model for Probabilistic Database Queries

Arthur H. van Bunningen<sup>1</sup>, Ling Feng<sup>2</sup>, Maarten M. Fokkinga<sup>1</sup>, Peter M.G. Apers<sup>1</sup>

<sup>1</sup>Centre for Telematics and Information Technology

University of Twente, The Netherlands

{bunninge, fokkinga, apers}@cs.utwente.nl

<sup>2</sup>Department of Computer Science & Technology

Tsinghua University, China

fengling@tsinghua.edu.cn

## Abstract

Following the availability of huge amounts of uncertain data, coming from diverse ranges of applications such as sensors, machine learning or mining approaches, information extraction and integration, etc. in recent years, we have seen a revival of interests in probabilistic databases. Queries over these databases result in probabilistic answers. As the process of arriving at these answers is based on the underlying stored uncertain data, we argue that from the standpoint of an end user, it is helpful for such a system to give an explanation on how it arrives at an answer and on which uncertainty assumptions the derived answer is based. In this way, the user with his/her own knowledge can decide how much confidence to place in this probabilistic answer.

The aim of this paper is to design such an answer explanation model for probabilistic database queries. We report our design principles and show the methods to compute the answer explanations. One of the main contributions of our model is that it fills the gap between giving only the answer probability, and giving the full derivation. Furthermore, we show how to balance verifiability and influence of explanation components through the concept of verifiable views. The behavior of the model and its computational efficiency are demonstrated through an extensive performance study.

## 1 Introduction

Since the pioneering work of Cavallo and Pittareli [4] research on probabilistic databases has focussed on delivering a general method for managing uncertain data. The first motivations for probabilistic databases were storing reliability of suppliers and statistics about customers. Nowadays, the availability of uncertain data coming from diverse ranges of applications, such as sensors, machine learning or mining approaches, information extraction and integration has increased enormously. This, together with

advances in query processing, has led to a recent revival of interest in applying probabilistic datamanagement systems [15].

Research in probabilistic databases has produced fruitful results, such as probabilistic relational algebra [7] and probabilistic query answering [15, 5], and recently the first implemented systems [11, 1] started to appear. Basically, a probabilistic relational database accommodates a set of relations, whose tuples are assigned probabilistic weights, indicating the probability that the tuple belongs to the corresponding relation. Queries over probabilistic databases will lead to probabilistic answers, reflecting the underlying uncertainty of tuples in their weights. An item being in the answer to a query is no longer a boolean value, but a probabilistic event.

Due to the lack of answer explanations and black-box styled query engines in the present-day probabilistic database systems, an inherent problem remains in the field. Because the query engine works with uncertain data, whose probability assumptions may be wrong in certain circumstances for certain users, the returned query result may contain incorrect answers, making the system questionable and unacceptable by its end users.

As an example, suppose an information integrator populates a relation *HASSTUDENT* (*groupname, student*) with two tuples: (*Utrecht, Harold*) of probability 80% and (*Utrecht, Sander*) of probability 20%. When a user asks for “*the names of the groups where Harold studies and not Sander*”, the query result will include answer *Utrecht* with the probability  $80\% \cdot (100 - 20\%) = 64\%$ . However, if this user is completely sure that Sander does work in *Utrecht*, *Utrecht* is apparently a wrong answer.

To resolve this problem, user’s interaction with the system is a must. It is desirable for the query system to open the black box and explain to the user how it arrives at a query result, and which uncertainty assumptions (probabilities) form the basis of the derived query result, so that the user can decide how much confidence to place in the answer based on his or her own knowledge. A first step in this direction is the work on lineage in probabilistic databases [2]. Nevertheless, astute readers may argue that the way of presenting all the base assumptions in the database related to resulting tuples may not be practical and user-friendly, particularly in situations where the user has (or wants) limited amount of time to examine the result, and the underlying uncertainty assumptions are overwhelming. In other words, the answer explanations provided by the system must be *concise* and *vital* enough, so that users can quickly and easily justify the query answer based on their own knowledge.

To fill the gap between giving only one answer probability (“*non-explanation*”), and giving the full answer derivation process (“*over-explanation*”), this paper presents an answer explanation model for probabilistic database queries.

The remainder of the paper is organized as follows. First, we review some closely related work on answer explanation in Section 2. The design of an answer explanation model for probabilistic database queries is detailed in Section 3. We show the ways to compute an answer explanation in Section 4 and study the performance in Section 5. We address some challenges induced by answer explanation and conclude the paper in Section 6.

## 2 Related Work

Work on explanation of answers stems from the field of logic. In the area of information systems, explanation takes two roles. The system either provides knowledge and explanations necessary for the user to carry out his or her task, or alternatively the system carries out some action and then explains the need and reason for the action that the system itself has taken to the user [9]. As a result, most of the explanation facilities were offered by those expert systems that used rule-based reasoning to arrive at conclusions. MYCIN [3] is one of the most exemplary system examples. It outlined five specific reasons for an explanation from an expert system perspective, namely, understanding, debugging, education, acceptance, and persuasion, and provided explanations by translating traces of rules followed from LISP to English. It implemented several explanation options, of which the two most important ones were: why the expert system asked the user a certain question, and how the system deduced a certain conclusion.

In the area of recommender systems [16], explanation mechanisms were also exploited to achieve the following five main targets: Transparency - explaining how the system works; Scrutability - allowing users to tell the system it is wrong; Trust - increase users' confidence in the system; Effectiveness - help users make good decisions; Persuasiveness - convince users to try or buy the recommended items; Efficiency - help users make decisions faster; Satisfaction - increase the ease of usability or enjoyment [16].

In addition, there have also been considerable studies into cognition, psychology, and philosophy of questioning and question answering with humans and how it can be applied to human-computer interaction [10, 8]. Johnson and Johnson [9] researched into the components of a unified theory of explanation and performed empirical experiments to help determine the logical components of an explanation.

In the database field, answer explanations are traditionally concerned with query plans and path selections. The purposes are mainly to tune applications to take advantage of indices, instead of explaining why an answer was returned. Recently, we see some interesting work done in the Trio system [11] on the lineage of an answer. Furthermore, the work on ordering the attributes of query results [6] can also be regarded more or less related to explanation, since the proposed method for determining which attributes to show relates to the attributes that can best explain the score or rank for a tuple.

## 3 An Answer Explanation Model for Probabilistic Database Queries

In this section, we first discuss the general principles that influence the design of our answer explanation model, and then present the designed model upon probabilistic databases.

### 3.1 Design Principles

Building an explanation facility can benefit systems and users in various ways, as described in the related work. The role of explanations determines the appropriate explanation model and associated quality measurement. As probabilistic systems work with probabilistic assumptions to derive probabilistic answers, both of which could be wrong, the explanation facility to be investigated in this study is thus aiming at *justification*. This bears most similarity to *understanding* and *debugging* from an expert system perspective, as well as *scrutability* from a recommender system perspective.

Considering that a user usually is more likely to recognize mistakes in basic assumptions leading to an answer, than mistakes in the answer itself, by revealing possibly wrong or correct assumptions used in answer derivation, the system can let the user, based on his or her knowledge about these assumptions, decide how much confidence to place in the final derived probabilistic answer. This explanation can also help enforce the acceptance of the system, since if the user understands why an answer is wrong, s/he would not question much about the behavior of the system.

For our purpose of justification, we defined the following four considerations during the design of our answer explanation model for probabilistic database queries.

1. *Verifiability*. In response to the justification requirement, the contents of an explanation provided must be verifiable by the users.
2. *Concision*. From a user's viewpoint, a long list of explanation contents for an answer is normally undesirable and overwhelming, particularly when the user has limited amount of time to examine the query result. Therefore, the explanation provided must be concise enough, standing in between the traditional no-explanation and over-explanation.
3. *Influenceability*. Closely related to the above concision requirement, the explanation contents offered must have the greatest influence upon the answer. That is, if these contents are changed, there is a high chance that the answer will be affected and changed accordingly. There is no use to explain something whose change makes no difference for the answer.
4. *Doubt*. The contents of an explanation preferably exhibit a certain degree of uncertainty. In other words, there is a relatively high chance that these contents may be wrong under certain circumstances. It makes no sense to explain something to the user, while the system is pretty sure about its correctness.

### 3.2 A Brief Review of Probabilistic Databases

In the literature, there exist two approaches for dealing with querying in a probabilistic database, namely, based on *extensional semantics* and *intentional semantics*.

The extensional semantics approach is to modify the operators of the algebra to compute probabilities. The problem of this approach is that it ignores most correlations between intermediate results and may give different results depending on the query plan [5].

The above problems are circumvented by the intentional semantics approach, which keeps track of the basic probabilities that contribute to a derived fact and determine the probability of this derived result [7]. The basis of an intentional probabilistic relational data model is to view each tuple as a probabilistic event. If the event is true, the tuple belongs to the relation, and otherwise not.

Furthermore, there is a distinction between *basic event* and *complex event*. Basic events are atomic events that do not depend on other events. Their probabilities (i.e., assumptions) are stored in the database. Relations that store tuples with basic events are called base relations. Tuples in these base relations are called *base tuples*.

From base relations, further relations can be derived by means of relational operators. These derived relations are also called views in the relational database field. A relational view contains derived view tuples, each of which corresponds to a complex event. Here, views can be either virtual or materialized. Through the basic events, probabilities of complex events can be computed. An *event expression* can be either a basic event or a complex event.

The intentional semantics approach is sometimes considered impractical, since the number of event expressions can become very large and calculating probabilities of the event expressions is an NP-complete problem. However, some effective optimization techniques have already been developed, such as exploiting independency between parts of the computation as suggested by Das Sarma *et al.* [13] and approximation using Monte Carlo algorithms as suggested by Dalvi and Suciu [5].

In this paper, we base our work on the well-established intentional probabilistic data model and its probabilistic relational algebra, as introduced by Fuhr and Rolleke [7]. Limited by space, we will not describe in detail the general derivation procedure from relational operators (SQL queries) to event expressions. For the exact details, we point to the work of Fuhr and Rolleke [7].

### 3.3 The Model

The answer to a probabilistic database query contains a list of base (or derived) tuples originating from the base relations. Each resulting tuple is attached with a probability, computed based on the probabilities of the base tuples.

It is common that a user can more easily recognize mistakes in underlying meaningful assumptions that lead to an answer than mistakes in the answer itself. In this study, we assume that a user is more likely to verify the truth of a base tuple or a meaningful view tuple than that of a derived answer. For the view tuples, we consider only views that are explicitly defined by the user as being verifiable.

Furthermore, suppose that each such tuple is equally understandable. Following the concision and influenceability considerations, the strength of the relationship between an explanation and an answer should be maximal. For example, a perfect tuple given as explanation should be true if the answer is true, and false if the answer is false.

Hence, we choose the base/view tuple which has the most extreme *correlation* with the answer as the explanation content. We measure the correlation between an answer tuple  $A$  and a base/view tuple  $T$  as follows.

$$\text{Corr}(A, T) = P(A \wedge T) - P(A) * P(T) \quad (1)$$

where  $P(A)$  and  $P(T)$  are the probabilities of tuple  $A$  and  $T$ . The correlation is most extreme when  $T$  and  $A$  are either identical or opposite.  $\text{Corr}(A, T) = 0$  implies that  $A$  and  $T$  are independent.  $\text{Corr}(A, T) > 0$  implies that  $A$  and  $T$  are positively correlated, meaning that an increase in the probability of either of them will also increase the probability of the other. Similarly,  $\text{Corr}(A, T) < 0$  implies that they are negatively correlated.

One might argue that even if a highly correlated tuple is wrong, it does not imply that the answer is definitely wrong (this happens only for tuple where  $P(A \wedge \neg T) = 0$ ). However, the tuples possessing the most extreme correlation with the answer enforce the most influence.

Another validation for using the correlation is that the tuple with the most extreme correlation “improves the probability” the most. We define the probability improvement as  $P(T) \cdot (P(A|T) - P(A)) + P(\neg T) \cdot (P(A) - P(A|\neg T))$ . In words; either the tuple is true or false. If the tuple is true, the confidence changes with  $P(A|T) - P(A)$ . If the tuple is false, the confidence changes with  $P(A) - P(A|\neg T)$ . Since the user can verify whether the tuple is true or false and if we suppose that evidences are not conflicting (which is the best we can do), the probability is improved with either  $P(A|T) - P(A)$  or  $P(A) - P(A|\neg T)$ , to the actual value which is  $P(A|\text{all evidence})$ . We can show that the probability improvement is equal to two times the correlation as can be seen in the following derivation:

$$\begin{aligned} & P(T) \cdot (P(A|T) - P(A)) \\ & \quad + P(\neg T) \cdot (P(A) - P(A|\neg T)) \\ = & P(T) \cdot \left( \frac{P(A \wedge T)}{P(T)} - P(A) \right) \\ & \quad + P(\neg T) \cdot \left( P(A) - \frac{P(A \wedge \neg T)}{P(\neg T)} \right) \\ = & P(A \wedge T) - P(A)P(T) \\ & \quad + P(A)P(\neg T) - P(A \wedge \neg T) \\ = & P(A \wedge T) - P(A)P(T) \\ & \quad + P(A) * (1 - P(T)) - (P(A) - P(A \wedge T)) \\ = & P(A \wedge T) - P(A)P(T) \\ & \quad + P(A) - P(A)P(T) - P(A) + P(A \wedge T) \\ = & P(A \wedge T) - P(A)P(T) \\ & \quad + -P(A)P(T) + P(A \wedge T) \\ = & 2 \cdot (P(A \wedge T) - P(A)P(T)) \end{aligned}$$

Let’s take a different angle to explicate why correlation directly addresses our con-

siderations on doubt of the tuple and influenceability:

$$\begin{aligned}
Corr(A, T) &= P(A \wedge T) - P(A)P(T) \\
&= -P(A)P(T) + P(A \wedge T) \\
&= P(A) - P(A)P(T) - P(A) + P(A \wedge T) \\
&= P(A) * (1 - P(T)) - (P(A) - P(A \wedge T)) \\
&= P(A) * P(\neg T) - P(A \wedge \neg T) \\
&= P(\neg T)(P(A) - \frac{P(A \wedge \neg T)}{P(\neg T)}) \\
&= P(\neg T)(P(A) - P(A|\neg T))
\end{aligned} \tag{2}$$

which is the multiplication of the doubt about the tuple, and the influence the tuple has on the answer.

In summary, our answer explanation model for probabilistic database queries is defined as follows.

**Definition 3.1** Let  $\mathcal{A}$  denote an answer, containing a list of resulting tuples with probabilities, i.e.,  $\mathcal{A} = \langle (A_1, E_1, P(A_1)), \dots, (A_n, E_n, P(A_n)) \rangle$ . Where  $E_i$  is the event expression for the derivation of answer tuple  $A_i$  according to the PRA-approach [7]. The **explanation** for answer  $\mathcal{A}$  is a list of explanations for the result tuples, i.e.,  $\Gamma(\mathcal{A}) = \langle \gamma(A_1, E_1, P(A_1)), \dots, \gamma(A_n, E_n, P(A_n)) \rangle$ . Each explanation component  $\gamma(A_i, E_n, P(A_i))$  ( $1 \leq i \leq n$ ) is for a resulting tuple  $A_i$  in  $\mathcal{A}$ , equal to  $(T_i, Corr(A_i, T_i))$ , where  $T_i$  is a base (view) tuple related to  $A_i$  satisfying  $T_i = \arg \max_T |Corr(A_i, T)|$   $\square$

### 3.4 Examples

To illustrate, we provide several answer explanation examples to a few representative queries. Table 1 is a small probabilistic database with 3 base relations, where the probabilities of the base tuples are assumed to be independent.

**Example 3.1** A query “look for the group where both Sander and Harold are studying” leads to the answer:

Result	Complex Event	Probability
Utrecht	$HS(UT,HA) \wedge HS(UT,SA)$	$0.8 * 0.2 = 0.16$

Intuitively, both tuples indicating that Sander and Harold study at Utrecht are equally important for the result, since in case that either Sander or Harold does not study at Utrecht, this answer will not be returned. However, for a conjunctive expression, the most influential one which could invalidate the answer is  $HS(UT,SA)$ , the most uncertain one. This is evidenced from the correlation which is for Harold  $P(\neg HS(UT,HA)) \cdot (P(Utrecht) - P(Utrecht|\neg HS(UT, HA))) = 0.2 \cdot (0.16 - 0.0) = 0.032$ , and for Sander  $P(\neg HS(UT,SA)) \cdot (P(Utrecht) - P(Utrecht|\neg HS(UT, SA))) = 0.8 \cdot (0.16 - 0.0) = 0.128$ .  $\square$

**Example 3.2** A query “look for the group where either Sander or Harold is studying” has the answer:



correlation of  $AW(LI,AC)$  with the answer is the highest. The system should therefore explain that Linda is an ACM Fellow for the result tuple. The interesting part of this example is that, in this case, the most important explanation for a returned tuple is not its direct attributes.  $\square$

Apart from incorporating base tuples into an answer explanation, it is possible to consider relevant view tuples as long as these view tuples are more meaningful and expressive, and thus more easily verified by users.

For example, a user may be able to verify the high-level concept of a “GROUP-SUPERVISOR” (a professor that supervises people for a group). We can derive such a view table as:

<b>GROUPSUP(group,prof)</b>	<b>Complex Event</b>
(Utrecht,Linda)	$((HS(UT,SA) \wedge SU(SA,LI))$ $\vee(HS(UT,HA) \wedge SU(HA,LI))$ $\vee(HS(UT,PA) \wedge SU(PA,LI)))$
(Eiland,Henk)	$((HS(EI,PE) \wedge SU(PE,HE))$ $\vee(HS(EI,PU) \wedge SU(PU,HE)))$

The correlation between a view tuple and a result tuple can be calculated in a similar way. To balance the influence, expressiveness, verifiability, and the associated computational complexity, it is reasonable to consider both base and view tuples as the candidate explanation components.

## 4 Computation of an Answer Explanation

### 4.1 Basic Approach

It is a challenge to compute the correlation between a base tuple and a result tuple, since calculating the probability of an answer tuple  $A$  (i.e., a complex event expression) is of exponential complexity, so does the calculation of  $P(A \wedge T)$  in E.q. (1).

To calculate the probability of an answer tuple, one can first convert its complex event expression into an equivalent disjunctive normal form, and then apply the following inclusion-exclusion (IE) formula [7]:

$$\begin{aligned}
 P(A) &= P(C_1 \vee \dots \vee C_n) \\
 &= \sum_{i=1}^n (-1)^{i-1} \sum_{1 \leq j_1 < \dots < j_i \leq n} P(C_{j_1} \wedge \dots \wedge C_{j_i})
 \end{aligned}$$

For each  $i$ , the formula picks all possible subsets like  $\{C_{j_1}, \dots, C_{j_i}\}$ , each containing  $i$  number of different elements from  $\{C_1, \dots, C_n\}$ , and sums their probabilities together. As base event expressions are assumed to be independent in this study,  $P(C_{j_1} \wedge \dots \wedge C_{j_n}) = P(C_{j_1}) * \dots * P(C_{j_n})$ . At each iteration, the obtained total amount is affiliated with a factor  $(-1)^{i-1}$ . Finally all the results ( $i = 1 \dots, n$ ) are summed together to get  $P(A)$ .

To reduce the overload due to result explanation, we propose to integrate the computation of the answer probability  $P(A)$  together with the correlation for each relevant base tuple  $T$ . Algorithm 1 details such a calculation procedure, where  $A$  is conveyed through an event expression in a disjunctive normal form  $(C_{1,1} \wedge \dots \wedge C_{1,x_1}) \vee \dots \vee (C_{n,1} \wedge \dots \wedge C_{n,x_n})$ , whose conjunct subset is  $\text{ConjunctSubset} = \{\{C_{1,1}, \dots, C_{1,x_1}\}, \dots, \{C_{n,1}, \dots, C_{n,x_n}\}\}$

---

**Algorithm 1:** Calculation of  $P(A)$  and  $P(A) - P(A|\neg T)$  for each related base tuple  $T$

---

**input** : An event expression (answer)  $A$   
**output:** The probability  $P(A)$  and a dictionary storing  $P(A) - P(A|\neg T)$  for each basic event expression (base tuple)  $T$

```

1 total_probability = 0
2 forall ConjunctSubset in CE do
3   uniqueEE =  $\emptyset$ 
4   forall conjunct  $\in$  ConjunctSubset do
5     uniqueEE = uniqueEE  $\cup$  conjunct
6   subsum =  $(-1)^{|\text{ConjunctSubset}|-1} \cdot \prod_{T \in \text{uniqueEE}} P(T)$ 
7   forall  $T \in$  uniqueEE do
8     dict = storage_add(dict, T, subsum)
9   total_probability = total_probability + subsum
10  return total_probability, dict

```

---

In the algorithm, by summing up the `subsum` for all subsets of conjuncts in which a tuple appears (line 6), we get the probability  $P(A) - P(A|\neg T)$ , which by multiplication with  $P(\neg T)$  results in the correlation (see E.q. (2)). Function `storageadd` is responsible for storing the (intermediate) results  $P(A) - P(A|\neg T)$  for each relevant base tuple  $T$  in a dictionary (line 8).

**Example 4.1** Suppose we want to compute if Harold has Sander or Pavel as a companion at Utrecht. This is equal to the expression:  $((HS(UT,HA) \wedge HS(UT,SA)) \vee (HS(UT,HA) \wedge HS(UT,PA)))$ . In this case, we present the algorithm with the set:

$$\{\{HS(UT,HA), HS(UT,SA)\}, \{HS(UT,HA), HS(UT,PA)\}\}$$

The `ConjunctSubset` now iterates over the following subsets:

$$\{\{HS(UT,HA), HS(UT,SA)\}, \{HS(UT,HA), HS(UT,PA)\}\}$$

$$\{\{HS(UT,HA), HS(UT,PA)\}, \{HS(UT,HA), HS(UT,SA)\}\}, \emptyset$$

For each subset, `uniqueEE` is assigned the set of unique event expressions in the subsets:

$$\{HS(UT,HA), HS(UT,SA), HS(UT,PA)\}$$

$$\{HS(UT,HA), HS(UT,PA)\}, \{HS(UT,HA), HS(UT,SA)\}, \emptyset$$

Next, we calculate the subsum for each subset:

$$\begin{aligned}
& (-1)^1 \cdot P(HS(UT,HA)) \cdot P(HS(UT,SA)) \cdot P(HS(UT,PA)) \\
& (-1)^0 \cdot (HS(UT,HA)) \cdot P(HS(UT,PA)) \\
& (-1)^0 \cdot (HS(UT,HA)) \cdot P(HS(UT,SA)) \\
& 0
\end{aligned}$$

Which added together results in the answer:

$$= -(0.8 \cdot 0.2 \cdot 0.6) + (0.8 \cdot 0.6) + (0.8 \cdot 0.2) = 0.544$$

The difference between the initial probability of Harold having Sander or Pavel as a companion at Utrecht, and the probability of the answer if we knew that Pavel did not study at Utrecht ( $P(\text{Companion}) - P(\text{Companion} | \neg HS(UT,PA))$ ) is equal to the sum of all subsets in which Pavel appears, which is  $0.48 - 0.096 = 0.384$ . And hence the correlation between this tuple and the answer is  $0.384 * 0.4 = 0.1536$ .  $\square$

The provided algorithm can be easily modified to cater for the correlation between the answer and a view tuple, by computing for each set of unique conjuncts also the view tuples it matches.

Because of the enumeration of possible conjuncts using `ConjunctSubset` in  $A$ , the algorithm has an exponential time complexity  $O(c \cdot 2^{\#d})$ , where  $d$  is the number of disjuncts in the  $A$  and  $c$  is the average number of conjuncts for a disjunct. A naive algorithm, that would for each tuple calculate  $P(A \wedge T)$  would have an overall time complexity of  $O(|\text{uniqueEE}| \cdot c \cdot 2^{\#d})$ .

## 4.2 Optimization

The bottleneck of Algorithm 1 lies in its exponential nature in computing the probability of a complex boolean event expression.

To optimize, we can first look at possible safe query plans, which can guarantee the correct computation of the probability. A safe plan can result in PTIME complexity for computing the answer probability and correlations between the answer and base/view tuples. However, when a safe query plan is impossible or infeasible (e.g., too ineffective compared to an unsafe plan), in the database literature, two types of optimization techniques are used, i.e., exploiting independency in sub plans [13], and approximation [5, 12]. As this study focuses on exact answers, we exploit the use of the independency method, developed by Sarma *et al.* [13].

Take a complex query “look for the database groups which contain members that are supervised by the person who gets some award” for example. Here, we can identify two independent parts of the query request, namely, which group is supervised by which person, and which person has which award (Figure 1).

Identifying the independent parts is non-trivial, and the reader is pointed to the work of Sarma *et al.* [13], for an approach based on lineage. In our example, it could be determined that the underlying base-tuples of the *Award* part and the *Supervise* part

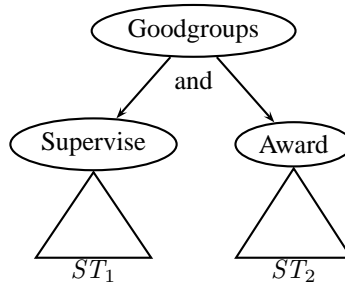


Figure 1: Independent parts of a query

are non-overlapping, and therefore independent.  $ST_1$  and  $ST_2$  in Figure 1 constitutes the two independent parts.

Exploiting the use of independent parts, we can optimize the computation (as shown by [13]). First, applying the computation upon independent parts, means that the exponential (e.g., IE) part only has to be applied to smaller subsets; e.g. the subtrees and their combinations instead of the whole tree at once. In our example, we can first compute the *Supervise* part (which has 4 disjuncts in the  $A$  and thus takes  $O(2^4)$ ), and then the *Award* part (2 disjuncts,  $O(2^2)$ ) instead of computing them during one time (8 disjuncts,  $O(2^8)$ ). Second, we can cache the probabilities of intermediate results ( $S$ ) from the independent subtrees. For example, if *Linda* also supervised members in a another university, we do not have to recompute the *Award*-part for the resulting probability.

The computation of correlation can be done similarly. That is, we not only store the intermediate probabilities, but also the most correlated tuple together with the probability  $Corr(S|\neg T)$ . The reason for this is that the tuple can be only correlated to the answer through  $S$ , and hence the tuple with the maximum correlation with  $A$  should also have the maximum correlation with  $S$ .

In our example, we combine the independent parts through the simple ‘AND’ operator. However, since such a combination can be arbitrarily complex, we use the IE-formula for computing the final probability and explanation component. When applying IE to the combination of the independent intermediate results (in Algorithm 1), for each intermediate result  $S$ , we store  $P(A|\neg S)$ . For example, for the intermediate result  $Supervise(Linda, Utrecht)$ , we store  $P(Goodgroup(Utrecht)|\neg Supervise(Linda, Utrecht))$ .

To calculate the correlation between  $A$  and a relevant base tuple  $T$ , we can use both the data stored per subset as well as the data stored for their combination. The key observation here is that since the tuple and the answer are independent given either  $\neg S$  or  $S$ , we can transform the correlation formula to a formula only dependent on  $P(A)$ ,  $P(A|\neg S)$ ,  $P(S)$ ,  $P(S|\neg T)$ , and  $P(T)$ , which are all stored in the system as can be seen if we rewrite the expression for  $Corr(A, T)$  as follows:

$$\begin{aligned}
Corr(A, T) &= P(\neg T)(P(A) - P(A|\neg T)) \\
&= P(\neg T)(P(A) - \frac{P(A \wedge \neg T)}{P(\neg T)}) \\
&= P(\neg T)(P(A) - \frac{P(A \wedge \neg T \wedge S) + P(A \wedge \neg T \wedge \neg S)}{P(\neg T)}) \\
&= P(\neg T)(P(A) - \frac{P(A \wedge \neg T|S) \cdot P(S) + P(A \wedge \neg T|\neg S) \cdot P(\neg S)}{P(\neg T)})
\end{aligned}$$

and because of the independences:

$$= P(\neg T)(P(A) - \frac{P(A|S) \cdot P(\neg T|S) \cdot P(S) + P(A|\neg S) \cdot P(\neg T|\neg S) \cdot P(\neg S)}{P(\neg T)})$$

Where:

$$\begin{aligned}
P(A|S) &= \frac{P(A \wedge S)}{S} \\
&= \frac{P(A) - P(A \wedge \neg S)}{S} \\
&= \frac{P(A) - P(A|\neg S) \cdot P(\neg S)}{S}
\end{aligned}$$

and

$$\begin{aligned}
P(\neg T|S) &= \frac{P(\neg T \wedge S)}{P(S)} \\
&= \frac{P(S|\neg T) \cdot P(\neg T)}{P(S)}
\end{aligned}$$

and

$$\begin{aligned}
P(\neg T|\neg S) &= \frac{P(\neg T \wedge \neg S)}{P(\neg S)} \\
&= \frac{P(\neg T) - P(\neg T \wedge S)}{P(\neg S)} \\
&= \frac{P(\neg T) - P(S|\neg T) \cdot P(\neg T)}{P(\neg S)}
\end{aligned}$$

In comparison with a base tuple, a view tuple can be either from an independent sub-tree like a base tuple or a combination of several sub-trees (intermediate tuple). For the latter, its correlation can be obtained by computing through  $P(A|\neg S)$ .

<b>MOVIE(id, title, rating, year)</b>		<b>ROMANCE(title, romance)</b>	
(M1, Shallow Grave, 7.4, 1994)		(Shallow Grave, 0)	
(M2, Shallow Hal, 6.1, 2001 )		(Shallow Hal, 1)	
...		...	
(a) Ground truth		(b) Original romance relation	
Event	Probability	<b>ROMANCE(id, romance)</b>	
RO(M1,RO)	0.73	(M1,1)	
RO(M2,RO)	0.73	(M2,1)	
...	...	...	
(c) Generated romance relation			

Table 2: Uncertain test data

## 5 Experiments

Two sets of experiments are conducted using Python (optimized with psyco<sup>1</sup>) on a 2.8Ghz Windows PC with 2GM RAM. The first experiment examines the behavior of our answer explanation model upon a probabilistic database, and the second one investigates the extra workload incurred by providing the answer explanation facility.

### 5.1 Experiment 1: The Behavior of the Model

#### 1) Test Data

We take a subset of the IMDB movie database<sup>2</sup>, containing 1632 movie records, each of which details the title, ranking, year, and romance genre of the movie. We simulate the data uncertainty by first splitting the table into two small tables. The first table has attribute id, title, rating, and year, and the second one contains attribute title, and romance genre, where value 1 indicates a romantic movie, and 0 otherwise. We then integrate these two small tables by title matching. To do this, we use the tri-gram distance and convert the distance into a posterior probability by assuming that it is normally distributed with mean 0, and variance  $\sigma$ , leading to a match-probability of  $e^{-distance^2/\sigma^2}$ , as described by Sen and Deshpande [14]. We further introduce noise by assuming that titles always have a distance deviation which is at least one.

As a movie may be matched to multiple movies, we normalize the probabilities for the romance genre per movie. This results in a target database of two relations. One certain relation (Table 2a) , and one with probabilistic statement about whether the genre of a movie is 'romance' (Table 2c).

Here, small values of  $\sigma$  lead to a certain database, while higher values of  $\sigma$  imply more uncertain data but still having a high variance. For a very large value of  $\sigma$ , the variance drops as all movies are mapped to each other and the probability for them being a romantic movie converges to  $\frac{\#Romantic\ movies}{\#Total\ movies}$ .

#### 2) Gain Measurement

<sup>1</sup><http://psyco.sourceforge.net/>

<sup>2</sup><http://www.imdb.com>

We measure the benefit (i.e., probability gain) of building the answer explanation facility as follows.

$$\begin{aligned} \text{gain} &= (P(A|T) \cdot P(T) + P(A'|\neg T) \cdot P(\neg T)) - P(A) \\ &= P(\neg T) \cdot (P(A'|\neg T) - P(A|T)) \end{aligned} \quad (3)$$

where  $A'$  is the best answer if the query system knew that the most correlated tuple  $T$  was false.  $A'$  can (or cannot) be equal to  $A$ . In other words, the user is able to verify the answer  $A$  based on the most correlated  $T$ , which can be either true or false. If  $T$  is true, the user will place more confidence in  $A$ . But if  $T$  is false, the user will favor the answer  $A'$ .

The gain measure has the following properties. First, from the comparison of E.q. (3) and the correlation in E.q. (2), and from the fact that for this experiment each tuple correlates either positively or negatively with all answers,  $P(A'|\neg T)$  can never be higher than  $P(A)$ , and therefore the gain is always bounded by the correlation. Furthermore, through the same comparison, as, in our experiment, tuples in the first answer do not influence the second answer (unless the answer is the same), we can see that, because  $P(A'|\neg T)$  is either  $P(A|\neg T)$  or  $P(A')$ , for the maximal correlation, also the gain is maximal.

### 3) Performance Results

We execute two sets of queries (disjunctive and conjunctive) over the generated probabilistic database (Table 2a, 2c) to investigate the behavior of our explanation model.

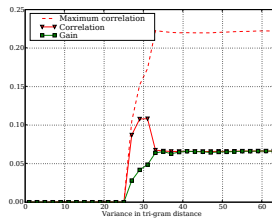
#### a) Disjunctive Queries

We first consider the disjunctive query: “look for the year after 1995 in which at least one of the top 5 rated movies was a romantic movie. Return the year of the highest probability”. This query request is made up of 5 disjuncts.

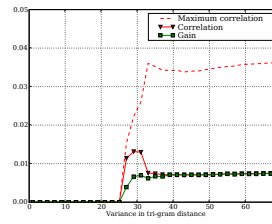
Figure 2a shows the results of the *maximal correlation* ( $=P(A) - P(A)^2$ , as stated in Section 3.2), the *correlation* between the answer and the base/view tuple in the explanation, and the *gain* under various values of  $\sigma$ .

For a year in which one or more movies were certainly romantic, the answer is always true and the correlation of base tuples with the answer is 0. Along with the increase of  $\sigma$ , the answer gets more uncertain and the maximal correlation rises. Eventually however the probabilities between the disjuncts become more equally distributed. The correlation therefore drops and converges to:  $(1 - P_{conv}) \cdot ((1 - (1 - P_{conv})^{\#\text{disjuncts}}) - (1 - (1 - P_{conv})^{\#\text{disjuncts}-1}))$ , where  $P_{conv}$  is  $\frac{\#\text{Romantic movies}}{\#\text{Total movies}}$ .

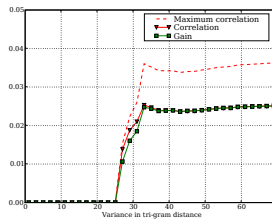
The above query was chosen in such way that the number of disjuncts is small (i.e., 5 disjuncts). How about increase the number of disjuncts in a query request, e.g., “look for the year after 1995 in which at least one of the top 15 rated movies was romantic movie”. Here the number of disjuncts is 15. From Figure 2b, we see a similar behavior, except for a much lower maximum possible correlation, correlation for explanation, and gain. This is because more disjuncts (15 in this case) contribute to the probability of answer  $A$ , resulting in a higher  $P(A)$ . Accordingly, the maximal correlation ( $=P(A) - P(A)^2$ ) becomes lower. Furthermore, each of the disjuncts enforces relatively a lower influence, leading to a lower tuple correlation bounded with  $A$ . However, when we



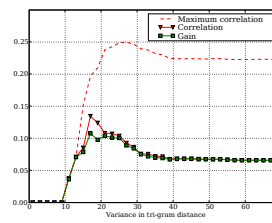
(a) Disjuncts, Top 5



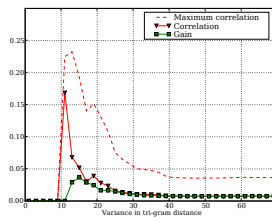
(b) Disjuncts, Top 15



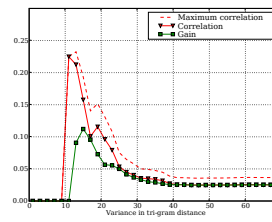
(c) Disjuncts, Top 15 with view



(d) Conjuncts, Top 5



(e) Conjuncts, Top 15



(f) Conjuncts, Top 15 with view

Figure 2:  $\sigma$  versus correlation and gain.

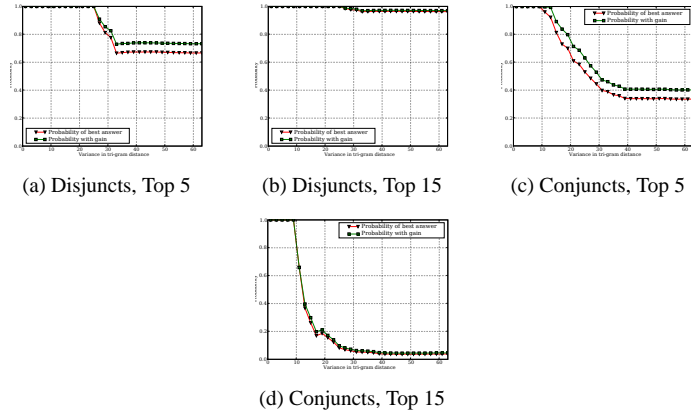


Figure 3:  $\sigma$  versus best answer probability (+ gain).

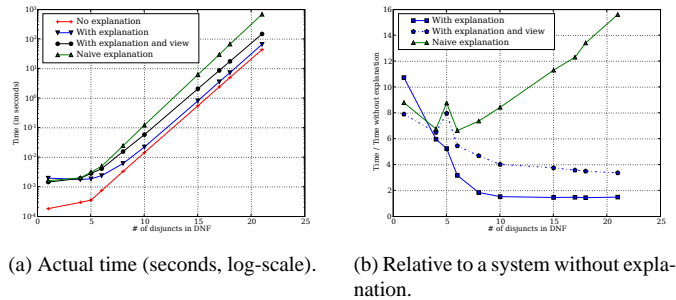


Figure 4: # Disjuncts versus running times for calculating probability and explanation.

shift to explain the answer via a view tuple instead of a base tuple, both the correlation for explanation, as well as the gain, all increase (Figure 2c). The reason is obvious, as a view tuple condenses a few base tuples together. As a whole, its correlation with the answer is bounded more tightly than individual tuples.

#### b) Conjunctive Queries

Next, we consider a conjunctive query: “look for all the years in which no romantic movie appeared in the top 5 (15) rated movies. Return the result of the highest probability”.

Figure 2d, 2e, and 2f exhibit a similar pattern as under the disjunctive queries, except that since an answer is wrong if only one of the conjuncts is wrong, we get an earlier rise in correlation when the database gets more uncertain. The effects of explainable views can be seen in Figure 2f, where we assume an explainable view which contains the 5 of the Top-15 movies that were the least likely to be romantic.

From the experiment, we are assured that presenting explanations does help to improve the confidence in query results, especially when the variance in data probabilities is high.

In the situations where there is either doubt about a few good answer options, or where there is a high probability for an answer which is still not absolutely sure about, explanations can serve as a kind of support in helping users to select the best answer from among the answer options, or justify the answer of high probability. In these cases, giving explanations to the user should be seriously considered. Furthermore, apart from base tuples, using more meaningful and condensed view tuples can enable the increase in both correlation and gain

## 5.2 Experiment 2: The Cost Incurred by the Model

To examine the performance loss incurred by providing answer explanations, we synthetically generate the query workload as follows. Let  $A = (C_{1,1} \wedge \dots \wedge C_{1,x_1}) \vee \dots \vee (C_{n,1} \wedge \dots \wedge C_{n,x_n})$  (where  $n$  varies from 1 to 22) be a returned answer in a normal disjunctive form, whose probability is to be calculated.  $x_1, \dots, x_n = 3$  in this test.

To provide an answer explanation, the most correlated base tuple and view tuple is returned based on the maximal correlation with  $A$ . Here, the view tuple is of the form  $(C_{1,1} \wedge C_{1,2}) \vee (C_{2,1} \wedge C_{2,2}) \vee (C_{3,1} \wedge C_{3,2})$ .

Figure 4a plots the times taken to compute  $P(A)$  with and without answer explanation. The relative times of explanation methods to a system without explanation is shown in Figure 4b. The naive explanation curve indicates the computing performance without using the optimization technique described in the previous section. As shown, the extra cost by adding answer explanation is small. This is because our strategy of integrating the probability calculation of the answer and the explanation together in one run.

## 6 Conclusion

While probabilistic databases serve as a good solution to manage the increasing amount of uncertain data available, the answers to probabilistic database queries remain probabilistic as well. We argue that building an answer explanation facility upon probabilistic databases is desirable to help users decide how much confidence to place in an answer according to their knowledge. In this study, we reviewed different goals of explanations in different areas, and decided a set of basic requirements for having answer explanations upon a probabilistic database system, aiming at answer justification. Following these requirements, we introduced an answer explanation model based on the correlation between an probabilistic answer to be explained and the explanation component, balancing verifiability and concision through the introduction of explainable view tuples. We integrated the computation of both an answer and its explanation together, so that the extra cost incurred due to explanation is the least, as evidenced by the experimental results. Furthermore, as evidenced from the experiments, our explanation method also lead to better query answers if a user takes into account the explanation in his/her choice.

Since answer explanation for probabilistic databases is still a rather new area of research, many challenges remain. We think it is usefull to categorize them in both user-oriented and system oriented challenges.

- *The user side:* In this paper, we chose justification as the goal for our explanations. This led to the use of using the most extreme correlation as an explanation primitive. As we discussed in Section 2, explanations can actually serve many other goals which may require different primitives. Furthermore, for the same goal, such as answer justification, the ideal explanation components could be different for different answering methods such as ranked answers [6]. *Defining motivated primitives* for different explanation goals is in our opinion a highly relevant area for future research. Besides, defining primitives goes hand-in-hand with *defining good answer presentations*. A good explanation might be useless if not presented adequately.
- *The system side:* Different kinds of explanation primitives call for different kinds of optimization techniques (e.g., answer approximation) to compute. Furthermore, primitives such as answer-feedback, might introduce possibilities for pre-computation of alternative results as well. In general, there are many possibilities in using motivated primitives for *optimization of computing answer explanations*.

We hope that we have convinced the readers of the benefits of providing explanation, and that it might be useful to support explanation primitives, such as correlation, at the system level. In this case, this research might be a small step towards more user-aware database systems.

## References

- [1] L. Antova, C. Koch, and D. Olteanu. Maybms: Managing incomplete information with probabilistic world-set decompositions. In *ICDE'07*, pages 1479–1480.
- [2] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. Uldbs: Databases with uncertainty and lineage. In *VLDB'06*, pages 953–964.
- [3] B. G. Buchanan and E. H. Shortliffe. *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.
- [4] R. Cavallo and M. Pittarelli. The theory of probabilistic databases. In *VLDB'87*, pages 71–81.
- [5] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB'04*, pages 864–875.
- [6] G. Das, V. Hristidis, N. Kapoor, and S. Sudarshan. Ordering the attributes of query results. In *SIGMOD'06*, pages 395–406.
- [7] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.
- [8] A. Graesser and J. Black. *Psychology of Questions*. Lawrence Erlbaum and Associates, 1985.
- [9] H. Johnson and P. Johnson. Explanation facilities and interactive systems. In *IUI '93: Proceedings of the 1st international conference on Intelligent user interfaces*, pages 159–166, 1993. ACM Press.
- [10] T. Lauer, P. E., and A. Graesser. *Questions and Information Systems*. Lawrence Erlbaum and Associates, 1985.

- [11] M. Mutsuzaki, M. Theobald, A. de Keijzer, J. Widom, P. Agrawal, O. Benjelloun, A. D. Sarma, R. Murthy, and T. Sugihara. Trio-one: Layering uncertainty and lineage on a conventional dbms. In *CIDR'07*, pages 269–274.
- [12] C. R. N. N. Dalvi, and D. Suciu. Query evaluation on probabilistic databases. *IEEE Data Eng. Bull.*, 29(1):25–31, 2006.
- [13] A. D. Sarma, M. Theobald, and J. Widom. Exploiting lineage for confidence computation in uncertain and probabilistic databases. Technical report, Stanford University, March 2007.
- [14] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *ICDE'07*.
- [15] D. Suciu and N. N. Dalvi. Foundations of probabilistic answers to queries. In *SIGMOD '05*, page 963.
- [16] N. Tintarev and J. Masthoff. A survey of explanations in recommender systems. In *Workshop on Recommender Systems and Intelligent User Interfaces associated with ICDE'07*.