Explaining Query Answers in Probabilistic Databases

Hichem Debbi*

Department of Computer Science, University of M'sila, M'sila (Algeria)

Received 8 August 2021 | Accepted 13 January 2023 | Early Access 24 July 2023

Abstract

Probabilistic databases have emerged as an extension of relational databases that can handle uncertain data under possible worlds semantics. Although the problems of creating effective means of probabilistic data representation as well as probabilistic query evaluation have been addressed so widely, low attention has been given to query result explanation. While query answer explanation in relational databases tends to answer the question: why is this tuple in the query result? In probabilistic databases, we should ask an additional question: why does this tuple have such a probability? Due to the huge number of resulting worlds of probabilistic databases, query explanation in probabilistic databases is a challenging task. In this paper, we propose a causal explanation technique for conjunctive queries in probabilistic databases. Based on the notions of causality, responsibility and blame, we will be able to address explanation for tuple and attribute uncertainties in a complementary way. Through an experiment on the real-dataset of IMDB, we will see that this framework would be helpful for explaining complex queries results. Comparing to existing explanation methods, our method could be also considered as an aided-diagnosis method through computing the blame, which helps to understand the impact of uncertain attributes.

Keywords

Causality, Conjunctive Queries, Explanation, Probabilistic Databases, Query Answers.

DOI: 10.9781/ijimai.2023.07.005

I. Introduction

RECENT applications in different domains are producing a huge volume of imprecise or uncertain data. Applications such as RFID and sensor networks are reporting frequently imprecise information, which is due mainly to measurement errors. In other applications such as information extraction from text or web pages, the extraction process yields automatically probabilistic results, due to the imprecise data source, or ambiguity in natural language text. A most known example is NELL [1], the Never Ending Language Learner that learns over time by reading the web [2]. It produces a set of facts, each one is assigned a probability score representing the confidence of the fact extracted. Business analysis, data cleaning, and integration are also quite active domains for dealing with uncertain data.

Due to the high importance of dealing with uncertain data, and since traditional databases do not have the ability to store and query uncertain data, probabilistic databases have emerged to address this issue. In probabilistic databases, the tuple may exist with a certain probability, or the values of some attributes may be uncertain [3]. We refer to both two types of uncertainty as *tuple-level uncertainty* and *attribute-level uncertainty*. For modeling both types, we use *the possible worlds semantics* [4]. It states that the real database is not known with certainty, therefore we introduce a probability distribution on all possible instances or worlds of this database.

Among major challenges in relational databases we find the following related problems: consistent query evaluation, data repairs, data cleaning, and explanation of unexpected query results. Consistent query evaluation (CQA) refers to computing meaningful answers to queries when dealing with an inconsistent database [5]–[7]. A database is considered inconsistent if it does not satisfy a set of specifications called integrity constraints. To restore consistency with regard to these constraints, different database repairs semantics have been proposed. The general idea is based on finding a consistent database close to the inconsistent one with a minimal number of repairs, and look for answers that are true in all repairs [8]. When we want to deal with this inconsistency we can employee diagnostic approaches that try to find the root causes for this inconsistency. Thus we face a matter of causality. Similarly, in data diagnosis or in data cleaning [9], we look generally for a set of causes. However, we face more a question of causality when we try to explain unexpected query results. In this regard, many approaches have been proposed leading to an interesting subject, which is causality query answering. These approaches are based mainly on analyzing the query result in the form of the query lineage. Lineage is a standard and powerful tool that helps us to track every output tuple in the query result to its origins or input tuples.

Although uncertain data representation as well as querying have been addressed so widely [4], [10]–[12], low attention has been given to query answer explanation comparing to classical databases. Kanagal and Deshpande [13] addressed this issue into two dimensions: the qualitative dimension, which refers to a classical question, why such an output tuple is in the query result, thus they try to identify the cause of the answer; the quantitative dimension: why does an output tuple have a high probability, thus they try to identify the input tuples' probabilities that significantly contributed to the output probability. Measuring such contributions is based on measuring the probability difference when altering the probabilities of these input tuples.

In probabilistic databases, depending only on lineage does not provide enough explanation, since the context is quantitative and the query result usually consists of multiple tuples. In this regard, Re and Suciu [14] proposed two approximate lineage techniques, sufficient

* Corresponding author.
E-mail address: hichem.debbi@univ-msila.dz

Please cite this article in press as:
H. Debbi. Explaining Query Answers in Probabilistic Databases, International Journal of Interactive Multimedia and Artificial Intelligence, (2023), http://dx.doi.org/10.9781/ijimai.2023.07.005

and polynomial lineages that provide a high compact representation of lineage that only takes into account the influencing set of tuples. Providing an efficient small lineage through these approximate techniques helps to track down any derivations and provides better explanations. While this work considers only conjunctive queries, Kanagal et al. [13] addressed in addition aggregation and top-k queries. One additional difference is that the first one addresses the creation of lineage, the later build their algorithms on top of the lineage formula, and then try to extract the most influential input tuples. In contrast to looking for previous approaches that look for causes, Ceylan et al. [15] have investigated the explanation of probabilistic queries with the aim of finding the most probable database and the most probable hypothesis for a given query. They studied these problems with respect to both conjunctive and ontology-mediated queries, and the complexity analysis results showed that it could be helpful for applications in prediction and diagnosis tasks.

Causality plays an important role in explaining any phenomena. Halpern and Pearl have introduced a causality model, which they refer to as structural equations [16], [17]. This model of causality has been successfully used in many research areas. Based on this definition, Halpern and Chokler [18] introduced the definition of responsibility. Responsibility extends the concept of all-or-nothing of the actual cause $X = x$ for the truth value of a Boolean formula ϕ in (M, u), where u refers to a context, and M refers to the causality model. It measures the number of changes that have to be made in u in order to make ϕ counterfactually depends on X . When we have an uncertainty around the context, we face in addition to the question of responsibility the question of blame.

In this paper, we employ these definitions: causality, resposnbility and blame to provide explanations for query answers in probabilistic databases. We take the lineage produced as input, and then try to identify the most responsible tuples and the uncertain variables that have the most blame for such an outcome. Our work for identifying and ranking causes with their degree of responsibility is similar to previous works [19], [20] in a way that it tries to identify the most responsible tuples for such an outcome. However, we address them in an uncertain setting, which leads us to propose an extended version of responsibility, which is probabilistic responsibility. The second part of our work aims to identify the attributes having the most blame. Although the uncertainty in probabilistic databases is mainly associated with uncertain attributes, to our knowledge, there has not been an attempt before to address the contribution of uncertain attributes.

To clarify the role of such diagnostic information, we use in our paper the form of U-relational databases [21] that are based on attribute-uncertainty. A U-relational database is featured in many probabilistic database management systems such as MayBMS [22]. MayBMS is considered one of the most successful probabilistic systems, which is built on top of an existing relational database management system [23].

Our technique is complete in a way that it could touch both forms of probabilistic databases (tuple and attribute uncertainty) in a complementary way. While probabilistic responsibility is used for measuring the contribution of most responsible tuples, this same measure is employed then to rank uncertain attributes with the most blame. To our knowledge, this is the first attempt to employ all these notions together: causality, responsibility and blame in probabilistic databases. Our work is similar to Kanagal and Deshpande [13] in a way that the algorithm proposed is built on top of lineage. To show the effectiveness of our approach, we conducted two main experiments. We first evaluate the execution time of the proposed algorithm by varying a number of parameters. This experiment has been done on synthetic data. Then, we show the usefulness of our approach on a real probabilistic database, which is the IMDB database [24].

In the following we summarize the main contributions of the paper:

- We provide a causal-based explanation framework for analyzing the query answers in probabilistic databases.
- It is the first time, where all the notions of causality, responsibility and blame are combined together in a synergistic way.
- By performing extensive experiments, we will show that our framework is scalable even for large databases, inducing millions of causes.
- Although we do not have enough available real probabilistic datasets for evaluation, we succeeded to get promising results by executing our framework on a real-dataset, which is the IMDB dataset. To our knowledge, explaining queries over IMDB dataset has been addressed for the first time.
- Our method does not act just as an explanation method for query answers, but also acts as an aided diagnostic method that helps to understand the contribution of uncertain attributes.
- The experiment on IMDB has been executed on the top of one of the successful probabilistic database management systems MayBMS [23].

The rest of this paper is organized as follows. In Section 2 we present some related works. We present some preliminaries and definitions in Section 3. We introduce U-relational databases as well as lineage in this section. In addition, we revise the definitions of causality and responsibility in relational databases. In Section 4, we present our definitions for causality, responsibility and blame in the context of probabilistic databases. This section is ended by introducing an algorithm for computing all the measures related to our definitions. Experimental results are presented in Section 5. The last section concludes the paper and outlines some future work.

II. Related Work

Meliou et al. [19], [25] have addressed causality in relational database for the first time based on the definition of causality by Halpern and Pearl [16]. Given a query output over a database instance, they try to find the responsible tuples that cause this answer. A tuple t is considered as a cause for a query answer, if there exists such a contingency that represents a set of tuples called endogenous, in way that removing this set from the database makes the query output counterfactually depends on t, i.e, removing t will lead to a non-answer. This definition has been related to lineage based on *c-tables* [26], [27]. The lineage formula is introduced in Disjunctive Normal Form (DNF), in which, every tuple is represented by a Boolean variable, then a cause is considered as a tuple associated with a Boolean variable that is included in a minterm of the lineage formula. This definition has been enhanced by another quantitative measure called responsibility, which measures the degree to which a tuple is considered as a cause [18]. Computing responsibility of a tuple t is based on the size of its contingency, where the tuple with the lower contingency is supposed to has the highest responsibility and vice versa. In an extended work [25], they introduced a careful complexity analysis of computing causality and responsibility in databases for both why so and why no (non-answer) causality. A non-answer query result refers to the question: why some tuples are missed from the output ? In this regard, Diestelkämper et al. [28] addressed this issue in the context of Big data, on queries of the data-intensive scalable computing (DISC) Appach spark. Despite the application context of Appach Spark, which is so novel and relevant, this work compared to previous works on missing answers, addresses nested data, which lead to rely on specific a nested data model and nested relational algebra for bags as a query language . However, this work just like previous works on missing answers analysis relies on lineage or provenance, specifically the why-not provenance.

Some reported open problems by [19] [25] have been addressed later by Salimi [29]. Among problems addressed are databases repairs and consistent query answering, abductive reasoning in databases, and the view-update problem. They argue that these famous problems in databases have strong connections to the definition of causality in databases. They showed that computing causes and their responsibilities can be considered as a consistent query answering problem, and in order to obtain repairs, they proposed algorithms for computing causes and their responsibilities for queries as unions of conjunctive queries [30]. Based on Hitting sets and vertex covers problems in hyper-graphs, they introduced more details on the complexity analysis of causality in databases, that is uncovering some complexity issues that have not been addressed by Meliou et al. [25]. They also showed the connection between query answer causality and abductive diagnosis as well as view-update problem, particularly, the delete-propagation problem where only tuple deletions are allowed from views [20]. Delete-propagation process tries to minimize the side-effect on views, thus, looking for a minimal set of tuples deletion. This issue has been related to a minimal contingency set of a causes to establish the connection between the two concepts. In addition, for establishing this connection, they adapted conditioning causality [31] that states that computing causes for an unexpected answer should be guided or conditioned on some prior knowledge of the correctness of another set of outputs. As a result, a measure for a tuple contribution to different outputs may be obtained. They also showed the connection between abductive diagnosis and query answering causality in the context of Datalog queries.

Another interesting work that employs causality and responsibility has been done by Lian and Chen [32]. They addressed the problem of *probabilistic nearest neighbor* (PNN), which is related the context of moving objects such as RFID, sensor networks and location-based services that introduce usually imperfect estimations of objects positions. In this work, responsibility is assigned to an object. This object is considered as a cause for other objects to be or not to be included in PNN query answers.

In addition to relational databases, causality and responsibility have been implemented in knowledge based systems. Mu [33] has addressed the inconsistency of knowledge bases through affecting a degree of responsibility for each formula, starting from the hypothesis that this responsibility should be explained from a causal perspective. In this setting, computing the degree of responsibility of a formula for inconsistency is based on identifying the minimal number of formulas that have to be removed from the knowledge base in order to break all the minimal inconsistent subsets not containing the formula, where the formula is declared not responsible if it does not belong to any minimal inconsistent subset.

In contrast to all previous works adopting lineage for both traditional and probabilistic databases, Miao et al. [34] proposed CAPE (Counterbalancing with Aggregate Patterns for Explanations) for explaining aggregation queries that does not rely on lineage or provenance at all. They consider that in the presence of outliers in data, relying on lineage only could be misleading. Therefore, they look for some patterns that hold on the data to provide explanations counterbalancing the user's observation. That is, outliers contradicting some pattern related to the user question might be easily identified.

III. Preliminaries

A. Probabilistic Databases

In probabilistic databases, a database instance could be in several states, where each state has a degree of uncertainty. That is, we could have several possible instances, called worlds, each of which has a probability. To model these instances under uncertainty we use the possible worlds semantics [4]. It states that a probabilistic database is represented as a finite set of possible worlds with some weights, and these weights sum up to 1. We refer to such a finite set of structures an *incomplete database* [2].

Let us fix a relational schema that consists of k relation names R_1, R_2, ..., R_k. We refer to an incomplete database as $W = \{W^1, W^2, ...,W^n\}$, where each $W^i = \langle R_1^i, R_2^i, \ldots, R_k^i \rangle$ is a database instance. Now we define a probabilistic database as follows:

Definition 1. ***Probabilistic Database***. *A Probabilistic database is a probabilistic space* $D = (\mathrm{W}, \mathrm{P})$ *over an incomplete database W, where* $P{:}W \rightarrow [0,1]$ *is a probability distribution function such that* $\sum_{W \in \boldsymbol{W}} P(W) = 1$.

In probabilistic databases, relations instances are supposed to be different from a world to another. We call a relation R_j certain or deterministic, if $R_j^1 = R_j^2 = \ldots R_j^n$. Relations are different in the way that each relation instance R_j in a world W^i contains different tuples from an instance of the same relation in another world. These tuples are considered as probabilistic events because we are not sure that they represent certain data. Therefore, we define for each tuple a probability called *marginal probability* or *confidence*. The probability of a tuple $t \in R_j$ is defined as follows:

$$P(t \in R_j) = \sum_{1 \le i \le n : t \in R_j^i} P(W^i) \tag{1}$$

The question that arises now is how to evaluate a query Q on a probabilistic database. For doing so, two semantics have been considered so far. The fist is *possible answer sets* semantics, where the query is evaluated on every possible world, this returns a set of tuples for each world. Since the representation of all answers in not practical, we instead use the second semantics, which is called *possible answers* [2]. As in the first semantics, the query is evaluated on all possible worlds, however, the result is returned as a list of tuples annotated with probabilities. We can say that such a tuple t is a possible answer to a query Q, if $\exists W \in \mathbf{W}$ such that $t \in Q(W)$. We can say that a tuple is certain if $\forall W \in \mathbf{W}$, $t \in Q(W)$. Given a query Q and a probabilistic database $D = (\mathbf{W}, P)$, the marginal probability of a tuple $t \in Q$ is $P(t \in Q) = \sum_{W \in \boldsymbol{W} : t \in Q(W)} P(W)$. That is, the marginal probability of a tuple t is computed by summing up the probabilities of worlds in which the tuple t is returned as an answer for the query Q. So, the possible answers for a query Q with their probabilities are represented as $Q(D) = \{(t_1, p_1),(t_2, p_2), ...\}$. We can easily notice that two variants of tuple answer semantics can be defined, possible and certain. These sets are defined as follows :

$$Q_{poss}(\mathbf{W}) = \{t | (t,p) \in Q(D), p > 0\} \tag{2}$$

$$Q_{cert}(\mathbf{W}) = \{t | (t,p) \in Q(D), p = 1\} \tag{3}$$

B. BID Database

Probabilistic databases can be obtained through two types of uncertainties, either on the level of tuples, this type is called *tuple-level uncertainty*, or on the level of attributes, and this is called *attribute-level uncertainty*. In the first type, a tuple is considered as a random variable, so given a database instance, we are not sure if this tuple really exists. In the second type, the values of specific attributes are uncertain for each tuple, that is, the attribute is considered as a random variable, its domain is all the values that the attribute may take. During the query evaluation process, attribute-level uncertainty is usually transformed to a tuple-level uncertainty. Based on these types of uncertainty, we have two types of probabilistic databases: tuple-independent database, where the tuples are independent probabilistic events, and block independent-disjoint probabilistic database, where the tuples are partitioned into blocks according to an uncertain attribute, such that

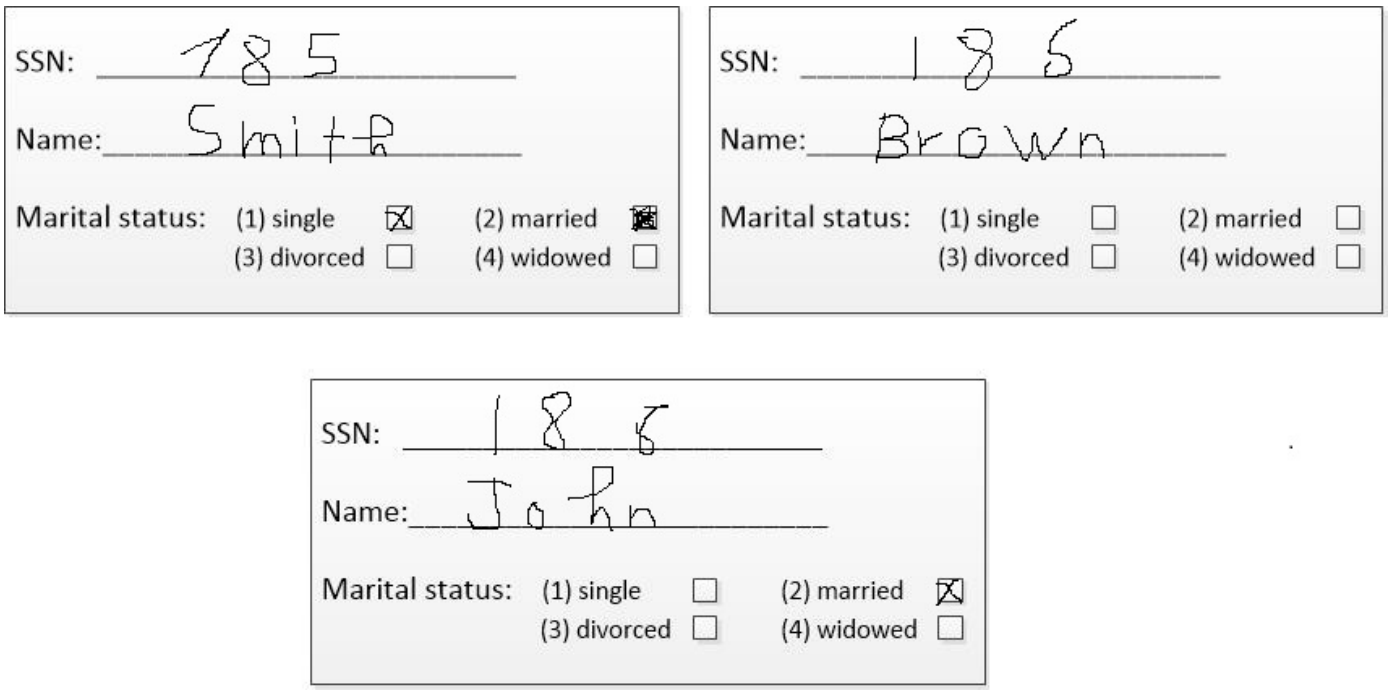

Fig. 1. Survey Forms.

tuples in the same block are disjoint and their probabilities sum up to 1, and tuples from different blocks are independent. So, in BID database, it is not possible for a world to contain more than one tuple from the same block. This representation is very effective and considered as a complete representation system with conjunctive query views [2]. This representation has been featured in many probabilistic database systems, such as MayBMS [22], [35], MystiQ [36] and Trio [37],[38]. It can also help for capturing key violations in databases.

Consider the forms presented in Fig. 1. It corresponds to 3 survey forms filled by three persons: Smith, Brown and John. Each form contains as information: social security number (SSN), name, and marital status (M). While both attributes ID and name have evident values, it is not the same case for SSN and M. For instance, one could be mistaken for the values of SSN in form 1, whether it is 185 or 785, and also in form 2 (185/186). One could be also mistaken for the value of M in form 1, weather its value is "single" or married", because it seems that Smith first checked "single" mistakenly, then he checked "married", but it could also be the contrary. In the second form, John did not check any status, hence, we have have four possible values. An example of BID database regarding these forms is presented in Fig 2. By taking exactly one value of each uncertain variable, this could result in $2 \times 2 \times 2 \times 3 = 24$ possible readings of the three forms. Suppose that we have millions of forms filled with this uncertainty, it would be a very challenging task to represent and process all these possible readings.

FId	SSN	P
1	185	0.4
1	785	0.6
2	185	0.7
2	186	0.3
3	186	0.75
3	188	0.25

FId	M	P
1	1	0.7
1	2	0.3
2	1	0.25
2	2	0.25
2	3	0.25
2	4	0.25
3	1	1

Fig. 2. An example of BID database.

C. U-Relational Database

U-relational database [11] is based on BID representation, and it is also considered as a probabilistic extension of classical conditional tables (C-tables) [39]. In a c-table, each tuple is annotated with a propositional formula over random variables. Using the logical operations *And* (∧), *OR* (∨) and *NOT* (¬), the propositional formula is obtained. In a U-relational database, the schema of each U-relation consists of: a tuple id column, a set of column pairs (V_i, D_i) that represent variable assignments or valuations, and finally a set of value columns. The probabilities of the assignments are stored in a separate table $W(V, D, P)$, called the world table.

Definition 2. ***U-relation Schema.*** *A U-relation schema is a represented as* $S = (V_1, D_1, ..., V_k, D_k, A_1, ..., A_m)$*, where k refers to the number of pairs of variable assignments (distinguished attributes), and m refers to the number of value columns. A U-relational database consists of U-relations.*

A U-relational database is an efficient and complete representation system that could provide a compact representation of the exponential number of possible worlds, and could also allow the representation of the result of any query [21]. Given U-relation database, the result of a query can be also returned in the same representation. A U-relational database for the example presented in Fig. 1 is presented in Fig. 3

In U-relational databases, each world W is defined by an assignment θ that assigns one possible value to each variable. Then, the probability or weight of this possible world is computed as the product of the probabilities associated to these valuations. For instance the probability of the world $W = \{x \mapsto 1, y \mapsto 3, z \mapsto 4, u \mapsto 1, v \mapsto 3, w \mapsto 1\}$ is $0.4 \times 0.3 \times 0.25 \times 0.7 \times 0.25 \times 1 = 0.0056$.

D. Lineage and Conjunctive Queries

Lineage is defined as propositional formula over input tuples in a database in order to explain how such an output query has been derived. Hence, each output tuple has a Boolean formula that states the input tuples responsible for its occurrence. Lineage is considered as a powerful tool for explaining query results. However, projection of million tuples on a single output tuple could result in a huge lineage formula. By considering uncertain context, i.e, probabilistic databases, the interpretation of lineage is more challenging. Lineage

$U_R[SSN]$	$V \mapsto D$	FId	SSN
	$x \mapsto 1$	1	185
	$x \mapsto 2$	1	785
	$y \mapsto 1$	2	185
	$y \mapsto 3$	2	186
	$z \mapsto 3$	3	186
	$z \mapsto 4$	3	188
$U_R[M]$	$V \mapsto D$	FId	M
	$u \mapsto 1$	1	1
	$u \mapsto 2$	1	2
	$v \mapsto 1$	2	1
	$v \mapsto 2$	2	2
	$v \mapsto 3$	2	3
	$v \mapsto 4$	2	4
	$w \mapsto 2$	3	2

$U_R[Name]$	FId	Name
	1	Smith
	2	Brown
	3	John

W	$V \mapsto D$	P
	$x \mapsto 1$	0.4
	$x \mapsto 2$	0.6
	$y \mapsto 1$	0.7
	$y \mapsto 3$	0.3
	$z \mapsto 3$	0.75
	$z \mapsto 4$	0.25
	$u \mapsto 1$	0.7
	$u \mapsto 2$	0.3
	$v \mapsto 1$	0.25
	$v \mapsto 2$	0.25
	$v \mapsto 3$	0.25
	$v \mapsto 4$	0.25
	$w \mapsto 2$	1

Fig. 3. An example of U-relational database.

in probabilistic databases is defined in a similar way comparing to relational databases, furthermore, query evaluation reduces to computing the lineage of output tuples, and then computing the probabilities of the lineage formulas.

Actually, all probabilistic database management systems use lineage-based query evaluation [12]. Many techniques have been proposed for computing the probability of propositional formulas in an efficient way, such as read-once formulas [40], OBDD [41], and d-DNNF [42].

Lineage formulas are written usually as DNF formulas. DNF formulas are Boolean formulas in disjunctive normal form. Formally, given a query Q and a probabilistic database D, each answer tuple $a \in Q(D)$ is associated with a DNF formula $\lambda_a^{Q,D}$.

In U-relational database systems like MayBMS, computing the probability or confidence of an output tuple is based on computing the probability of a DNF of this tuple, by summing up the probabilities identified with the valuation θ of random variables such that DNF becomes true under θ. Some approximation techniques based on Monte Carlo simulation have been proposed to approximate DNF probabilities computation [4], [43]. One interesting point about U-relational databases, is that a lineage of output tuples can be written in k-DNF formula for small number of k, where k represents the maximum number of literals.

Conjunctive query. Conjunctive query is the simplest and most used query in relational databases. It is restricted to the operators $\exists$ and $\wedge$ and it has the following form in relational calculus: $x, y \mid \exists z (R(x, y) \wedge S(y, z))$ where R and S are two relations. In Datalog, it is given in the form: $q(x, y) :- R(x, y), S(y, z)$. In SQL, conjunctive queries correspond to *selec − project − join*, and the *where* clause contains only equalities. By relating conjunctive queries to lineage, in SQL we can use: *select distinct attributes*, or alternatively the following form: *select * order by attributes*.

The lineage of an output tuple of a query in the first form would be a DNF formula whose terms are given by the rows returned by a query in the second form.

Example III.1. *Consider the U-relational database in Fig. 3. Consider a conjunctive query on two uncertain relations $U_R[SSN]$ and $U_R[M]$ that returns all possible SSNs of married persons (select SNN from U_SSN, U_M where U_SSN.FID = U_M.FID and U_M.M=2). Before going through query results, we should mention that some resulting worlds of this database could be inconsistent, in way that one world could contain two persons with the same SSN, which is not possible. Database management systems like MayBMS offers the possibility to detect such violations. An example of a query that repairs this database is given as the following:*

repair key (fid) in Census_SSN weight by p;

Where Census_SSN refers to the relation URrSSNs. After applying this constraint, we can reduce the number of instances of URrSSNs to four possible instances. Although the number of possible worlds is obviously more than 4, because we have another uncertain relation in hand, which is $U_R[M]$, we fix four instances as well for this relation to better explain our future notions. That is, we consider only the following 4 worlds:

$$W_1 = \{x \mapsto 1, y \mapsto 3, z \mapsto 4, u \mapsto 2, v \mapsto 2, w \mapsto 2\}$$
$$P(W_1) = 0.4 \times 0.3 \times 0.25 \times 0.3 \times 0.25 \times 1 = 0.0022$$
$$W_2 = \{x \mapsto 2, y \mapsto 1, z \mapsto 4, u \mapsto 1, v \mapsto 2, w \mapsto 2\}$$
$$P(W_2) = 0.6 \times 0.7 \times 0.3 \times 0.7 \times 0.25 \times 1 = 0.022$$
$$W_3 = \{x \mapsto 2, y \mapsto 3, z \mapsto 4, u \mapsto 2, v \mapsto 1, w \mapsto 2\}$$
$$P(W_3) = 0.6 \times 0.3 \times 0.25 \times 0.3 \times 0.25 \times 1 = 0.0033$$
$$W_4 = \{x \mapsto 2, y \mapsto 1, z \mapsto 3, u \mapsto 1, v \mapsto 1, w \mapsto 2\}$$
$$P(W_4) = 0.6 \times 0.7 \times 0.75 \times 0.7 \times 0.25 \times 1 = 0.055$$

For a query that returns married persons ($M = 2$), the query returns over these 4 worlds 8 possible tuples. Thus, the lineage formula would be a DNF formula consisting of 8 terms returned by this query over 4 possible worlds

$$[(x = 1 \wedge u = 2) \vee (y = 3 \wedge v = 2) \vee (z = 4 \wedge w = 2)] \vee$$
$$[(y = 1 \wedge v = 2) \vee (z = 4 \wedge w = 2)] \vee [(x = 2 \wedge u = 2) \vee$$
$$(z = 4 \wedge w = 2)] \vee [(z = 3 \wedge w = 2)]$$

The output probability is computed with respect to this DNF formula, which is the sum of these probabilities: 0.0825.

E. Causality and Responsibility in Relational Databases

Counterfactual reasoning that states: event A is a cause of event B if, had A not happened then B would not have happened, plays an important role in causality. Halpern and Pearl [16] extended this basic statement by taking A to be a cause of B, if B counterfactually depends on A under some contingency. Following this definition, Meliou et al. have introduced the definition of database causality. Let us fix a relational schema that consists of k relation names $R_1, R_2, ..., R_k$. Given a database instance D and a conjunctive query Q, for each relation R_i, we denote by R_i^n the set of endogenous tuples, and R_i^x the set of exogenous tuples. Endogenous tuples are those that are considered to be causes, whereas exogenous tuples are deemed not to be possible causes. That is, the tuples in D are partitioned into two sets $D = D^n \cup D^x$, where D^n, D^x represent all endogenous and exogenous tuples respectively. A tuple $t \in D^n$ is said to be a counterfactual cause for an answer a to Q in D, if $D \models Q(a)$ and $D - \{t\} \not\models Q(a)$. Given this definition, we can now give the definition of actual cause.

Definition 3. ***Actual cause***. *A tuple t is an actual cause for an answer a in D if there exists a set of endogenous tuples $\Gamma \subseteq D^n$, such that t is a*

counterfactual cause for a in $D - \Gamma$. We call this set a contingency for t.

That is, it is sufficient to find a set of tuples Γ that can be removed in order to make the query answer couterfactualy depends on the existence of *t*. In other words, removing t will switch to non-answer. It is evident that every counterfactual cause is an actual cause by taking $\Gamma = \emptyset$. Based on c-tables representation, computing the causes has been related to lineage in DNF formula [19]. Let us assume that every tuple t in D is associated with a Boolean variable X_t, and we denote by X^n the Boolean variables related to endogenous tuples. The DNF formula $\lambda_a^{Q,D}$ for such an output tuple a will consist only of X^n. In terms of lineage, we say that a tuple t is an actual cause for an answer a to a query Q on D, iff there exists a minterm in $\lambda_a^{Q,D}$ that contains t.

An open question has arisen given the above definition concerning a contingency Γ: does it matter the size of Γ on the importance of an actual cause? Chockler and Halpern have addressed this issue by introducing a quantitative measure called responsibility [18]. In relational databases responsibility has been used to rank tuples [19], [20].

Definition 4. ***Responsibility***. *The degree of responsibility of a cause t for an answer a of a query Q on D, denoted drt , is $dr_t = 1/(|\Gamma| + 1)$, where Γ represents the minimal contingency set for t.*

That is, the degree of responsibility of a tuple t is based on computing the number of tuples that we need to remove from the database D in order to make a query answer a contractually depends on t. Obviously, when the actual cause t is already a counterfactual cause, i.e, $\Gamma = \emptyset$, then the degree of responsibility $dr_t = 1$.

Let us consider a simple conjunctive query q_1 (*z*): $-R(x, y), S(y, z)$ over a database that contains the tuples $R(a, b), S(b, d), R(a, c), S(c, d)$. The lineage for the answer d would be $(X_{R(a,b)} \wedge X_{S(b,d)}) \vee (X_{R(e,c)}) \wedge X_{S(c,d)})$. It is evident that every tuple here is an actual cause for this answer. Removing $R(a, b)$ or $R(e, c)$ for instance will make both $S(b, d)$ and $S(c, d)$ counterfactual causes. It is evident that the minimal number of tuples needed for making these causes counterfactual is 1, that is each tuple here has a degree of responsibility 1/2. Now let assume the q_2 (*y*): $-R(x, y), S(y)$ over a database that contains the tuples $R(a, c), R(b, c), S(c)$, the lineage for the answer c would be $(X_{R(a,c)} \wedge X_{S(c)})$ $\vee$ $(X_{R(b,c)} \wedge X_{S(c)})$. Here the tuple $S(c)$ has a 1 as a degree of responsibility since its removal will result in no answer, whereas both of the tuples $R(a, c)$ and $R(b, c)$ share the responsibility.

IV. Causality and Responsibility in Probabilistic Databases

It is evident that causality and its quantitative extension responsibility play an important role in explaining query results in relational databases. Since probabilistic databases extend relational databases with probabilistic semantics, a great effort has been done mainly to extend relational semantics to represent uncertainty in data [2]. In this regard, we choose to extend the definitions of causality and responsibility to probabilistic databases, in order to explain the probabilistic results of a query over a probabilistic database. In addition, we use the definition of blame [18] that we consider as a helpful tool when we want to go deeply for explaining such results. Kanagal and Deshpande [13] have introduced some fundamental notions for explaining probabilistic results over probabilistic databases. They addressed questions like: why such a tuple is included in the result? in addition to: why a tuple t has more probability than the probability of another tuple t'? Some works try to enhance the computed lineage itself by proposing the notion of *approximate lineage*, which is considered to include only the most important and influential tuples. Here, we rely on complete lineage, and since it could be very huge, we employ these two definitions (causality and responsibility) to guide the user to the most responsible causes for such an output.

We should recall that probabilistic databases can be given into two representations: tuple-based and attribute-based. The approach that we are going to propose can handle both sources of uncertainties. While causality and responsibility can be used for explaining the first type, we count on blame for understanding the contribution of uncertain attributes to the probabilistic result of the query. So, we depend on causality and responsibility for answering questions like: Why is this tuple? what is its responsibility? what contribution does it have on the probability of the query? On the other side, we depend on blame for answering the question: *What uncertain variable we should blame the most for such an outcome*?. While the first question has been considerably addressed, to our knowledge, the second question has not been addressed before, though it is of high importance and could return deeper explanations. Furthermore, it could be very helpful for understanding the entire design of probabilistic databases.

Here we define causality, responsibility and blame by considering the probabilistic database to be U-relational database. As we showed before, U-relational databases are a complete representation, effective in many ways, and more importunately, it explicitly features attribute-uncertainty, which will be a ground for our definitions. One more important feature of U-relational databases is that causes can be returned in a detailed manner, i.e, in a form of valuations of the uncertain variables, this could provide a better explanation than an entire tuple that could consist of many attributes.

We introduce first the notions of causality and responsibility in general semantics, the possible worlds semantics. Hence, our definitions would be applicable for any probabilistic database, then we restrict our presentation with an example to U-relational database.

We should recall that computing the probability of a query output in U-relational databases is based on computing the probability of a DNF, which is the sum of the weights of the worlds identified with valuations θ such that the DNF becomes true under θ [44].

A. Causality and Responsibility

Before introducing our definitions, let us recall first what is a causal model according to Halpern and Pearl [16], on which the definitions of causality, responsibility and blame are built. A causal model is a tuple $M = (U, V, F)$, where the set U represents exogenous variables, whose values are determined by factors outside the model M, but they are necessary to encode the context, and the set of endogenous variables V, whose values are determined by a set of functions F. The causes are determined then by the valuations of V. The causality model can be extended to a probabilistic causal model as a tuple (M, Pr), where M is the causality model, and Pr is a probability function over possible contexts [17].

To relate this definition to probabilistic databases, the context will refer to a possible world W, whose probability is defined by the probability function P. V will refer to endogenous tuples in this world, this set is included in the DNF formula. U will refer to exogenous tuples that are not included in the DNF formula, but they are necessary to define the possible world. By considering U-relational databases, F will refer to the valuations θ.

Now we can introduce the definition of a cause in probabilistic databases. Let us consider a U-relational database D. We should recall that a U-relation schema consists of variables assignments (V_i, D_i), such that every tuple is associated with a valuation $V_i = v_i$. We recall also that given a query Q and a database D, each answer tuple $a \in Q(D)$ is associated with a global DNF formula $\lambda_a^{Q,D}$. Let us denote by $\lambda_a^{Q,W}$ the lineage for a in the local level of a world W.

Definition 5. ***Actual cause***. *A tuple t with a valuation $V_i = v_i \in \lambda_a^{Q,W}$ is an actual cause for an answer a in a possible world W, subsequently*

in D, if there exists a subset of variables $\Gamma \subseteq V$, *such that switching their values makes the truth value of* $\lambda_a^{Q,W}$ *counterfactually depends on* $V_i = v_i$.

That is, this definition is the same for a cause in relational database, just by considering one possible world. V refers to the variables associated with endogenous tuples, and $V_i = v_i$ is a valuation associated with an input tuple.

Definition 6. ***Responsibility***. *The degree of responsibility of a tuple t with a valuation* $V_i = v_i$ *for an answer a in a possible world W, is* $dr_t(V_i = v_i)^W = 1/|\Gamma|+1$, *where* Γ *represents the minimal contingency set for t.*

Conjunctions in lineage represent join over relations, that is, a valuation could appear in different conjunctions.

Proposition IV.1. Let $C_W = C_1, ..., C_n$ be the set of conjunctions related to DNF formula $\lambda_a^{Q,W}$. A tuple t with a valuation $V_i = v_i$ is a counterfactual cause, and thus it has responsibility 1, iff t is included in every conjunction C_i.

That is the cause with the highest responsibility will be the one appearing the most in the conjunctions of $\lambda_a^{Q,W}$.

Although, a cause responsibility is a very interesting measure, in this simplest form it did not yet take into account the uncertainty introduced by random variables V. Therefore, we have to deal with this uncertainty. Halpern and Pearl [17] addressed also the case where the context is uncertain, then the probability of a cause $X = x$, where $X \in V$, is given by the probability of the context u on which X has its value. We adopt the same definition here for cause in probabilistic database, and we define the cause probability in each world W as:

$$Pr_t(V_i = v_i)^W \stackrel{def}{=} P(W) | t \in W \tag{4}$$

That is, we have associated for each cause a probability that represents exactly the probability of the world in which it is included. Let us now denote by $W_a^{Q,D} \in W$ the set of worlds related to the global DNF formula $\lambda_a^{Q,D}$. Let a tuple t with a valuation $V_i = v_i$ be a cause that appears in a set of worlds $W_t \subseteq W_a^{Q,D}$.

Given the definitions of cause responsibility and cause probability, we introduce the following definition.

Definition 7. ***Probabilistic responsibility***. We define for each cause for an answer $a \in Q(D)$ a probabilistic responsibility as follows:

$$drP_t(V_i = v_i) = \sum_{W \in W_t} P\, r_t(V_i = v_i)^W \times dr_t(V_i = v_i)^W \tag{5}$$

That is, we define for each cause a responsibility over possible worlds, where each cause takes the probability of the world in which it is included. This has been done by computing the product of the cause responsibility and cause probability on the local level for every world, then summing up these measures. We consider this measure as an enhanced and required version of classical responsibility.

Definition 8. ***Most responsible cause***. A tuple t with a valuation $V_i = v_i$ is a most responsible cause for an answer $a \in Q(D)$, if $drP_t(V_i = v_i) \geq drP_{t'}(V_{i'} = v_{i'})$.

Consider for instance a world where we have a cause with responsibility 1. According to the classical definition, this cause represents a good explanation, however, when we know that the probability of this world is very low, the cause automatically loses its importance. Therefore, most responsible causes are those having high responsibilities in worlds with high probabilities. Obviously, we are interested more in most probable worlds. We should note here that a most probable database is supposed to be the closest to a certain database. It is evident that a most responsible cause does not have necessarily the highest probability, and it might not be unique.

Proposition IV.2. *A cause t could have* $drP_t(V_i = v_i) = 1$ *for an answer* $a \in Q(D)$, *which is the highest value possible for probabilistic causality, iff* $t \in Q_{cert}(\mathbf{W})$, *and t is a counterfactual cause.*

Proof. A certain tuple $t \in Q_{cert}(\mathbf{W})$ is defined as a tuple that is present in every world $W \in \mathbf{W}$ -See equation (3)-, i.e, $P(t) = 1$, which represents the sum of all worlds probabilities. With respect to Definition 7, the probabilistic responsibility of a tuple t (drP_t) is obtained by the product of its probability (P_t) and responsibility dr_t, so a cause probability is weighted by its responsibility. So, if this tuple t is a counterfactual cause in every world, then it has always 1 as a degree of responsibility, and since $t \in Q_{cert}(\mathbf{W})$, then the probabilistic responsibility of t will represent exactly the sum of all worlds probabilities, which is certainly 1, i.e, $drP_t(V_i = v_i) = 1$.

B. Blame

Blame has been proposed as a complementary notion for responsibility in the presence of uncertainty [18]. Actually, blame is used when responsibility does not give enough explanation, one source for this incompleteness is the uncertainty associated with the contexts in which responsibility is measured. As presented before, causality and responsibility have been proved to be a helpful tool for explaining such an answer, but we still need to question the probabilities derived. How uncertain attributes have contributed to this result, more deeply, which uncertain variable has the most blame for such an outcome. This question has more importance with conjunctive queries involving many uncertain relations.

Definition 9. ***Blame***. *Let X be an uncertain attribute, and* $U_R[X]$ *the uncertain relation. Let us denoted by* X_{TW} *the set of causes with valuations* $V_i = v_i, ..., v_m$ *with respect to X in a world W. We define for each uncertain attribute X for an answer* $a \in Q(D)$ *the degree of blame as follows:*

$$db(X) = \sum_{W \in W_t} \prod_{t \in X_{TW}} P(V_i = v_i) dr_t(V_i = v_i)^W \tag{6}$$

So, the degree of blame for an uncertain attribute X is based on computing first the products of the tuples probabilities associated with it in the world table, and their responsibilities with respect to each world, then summing up all these measures. That is we are trying to quantify the probability contribution of each cause valuation to the probability of the world taking into account its responsibility. We can say that we obtain here two diagnostic information, the first concerns tuples (causality and responsibility), and the second concerns the uncertain attributes (blame), where the second notion is based on the former notions.

Theorem IV.3. *If a tuple t with a valuation* $V_i = v_i \in X_{TW}$ *is a most responsible cause, it does not mean necessarily that X has the most blame.*

Proof. We can think of responsibility as an adjusting factor for both probability of a tuple P for computing the blame, and probability of cause Pr for computing the probabilistic responsibility. Let us consider that a cause t with a valuation $V_1 = v_1$ is included in the set of certain tuples $Q_{cert}(\mathbf{W})$, that is according to proposition IV.2, this is a most responsible cause with the highest possible value. It is sufficient to find an uncertain attribute $X2$ such that $db(X2 >) db(X1)$. We should notice that the most responsible cause when it comes to blame, is affected by the rest of valuations in X_{TW}. That is, it can be found that $X1_{TW}$ has low probabilities as well as low responsibilities compared to the valuations $X2_{TW}$, where $X2_{TW}$ does not include the most responsible cause.

C. An Algorithm for Explaining Query Answers

Our algorithm takes the lineage of such an answer in DNF form, and then returns the causes ordered with their probabilistic responsibilities, and the attributes ordered with their blame. However, before analyzing the DNF formula, some pre-processing is performed in order to compute the results in an efficient way. We propose to transform the DNF formula into a table of two dimensions, the first dimension represents the local lineage formula with respect

to each world, and the second represents the possible worlds. This representation provides all the information that we need to compute responsibility and blame. We call this representation a causality matrix.

The algorithm 1 aims to generate causes with their dR and Pr, and attributes with dB. To do so, we should introduce three main functions representing the main steps in computing probabilistic responsibility and blame. The objective of the first function *ComputeCauses* is to extract the causes from each term in the DNF formula, with respect to each world, the result will seem like a table of two dimensions. This table will contain redundant causes that appear in multiple terms, however, we need it to start the next phase. The second phase takes the result of the first phase as input, and tries to compute for each distinct cause its responsibility. This requires the computation of the contingency Γ for each cause. Based on the input, this is can be easily achieved through the use of a variable *K*. When the cause is not contained in a minterm, *K* is increased by one, which means that this term could be removed without affecting the truth value of $\lambda_a^{Q,D}$. Similarly, if the cause is contained in every term, thus, $K = 0$, it means that this cause is counterfactual and has responsibility 1. By completing the two loops, each cause will have its related responsibility. After that, we can compute the probabilistic responsibility of each cause. By completing this phase, we will have all required information in hand(causes with their *dR*, *Pr* and *P*) to compute the blame.

Algorithm 1 Compute Explanations

1: **Inputs**: DNF formula $\lambda_a^{Q,D}$
2: **Outputs**: Causes with responsibilities, attributes with blame
3: Causes = ∅
4: Causes = **ComputeCauses**($\lambda_a^{Q,D}$)
5: **ComputeRespons**(Causes)
6: **ComputeBlame**(Causes with dr and Pr)
7: **OutputExplanations**(Causes with dR and Pr, Attributes with dB)

It is evident that this algorithm returns the set of explanations in polynomial time. Its complexity depends on the size of *D* and the size of the DNF formula. For computing responsibility, since the DNF formula is related to a linear conjunctive query, computing responsibility has been already proved that it can be achieved in polynomial time [25], [45]. They have proposed a careful analysis of complexity for both causality and responsibility in relational databases, for both *Why so?* and *Why no?* causality.

Example IV.1. *Let us consider the results of the previous example of the census database (Example III.1). We want now to run our algorithm and measure the contribution of each cause to the previous output. We compute tuples responsibilities, as well as blame for the uncertain attributes SSN and M. We need first to perform a pre-processing on the lineage formula and construct the causality matrix. The causality matrix related to the lineage formula is presented in Fig. 4. This representation could help us to get a deep and complete insight on the causes and their probabilities, and help us to compute responsibility and blame so easily. Each column represents a cause with respect to each local lineage formula, double vertical lines refer to conjuncts of the local lineage formula. Computing responsibility is based on computing the number of conjuncts that do not include the cause. Let us take the cause $z = 3$, $z = 4$ and $w = 2$, and compute their responsibilities along the 4 worlds.*

$dr_t(w=2)^{W_1} = dr_t(z=4)^{W_1} = 1/3$
$dr_t(w=2)^{W_2} = dr_t(z=4)^{W_2} = 1/2$
$dr_t(w=2)^{W_3} = dr_t(z=4)^{W_3} = 1/2$
$dr_t(w=2)^{W_4} = dr_t(z=3)^{W_4} = 1$

In the first world $dr_t(w=2)^{W_1} = 1/3$ *since there exists two other conjuncts that do not contain* $w = 2$*, that is, in order to make* $w = 2$ *counterfactual cause, there exists two other contingencies. In* W_2 *and* W_3*, there exists one contingency,* $dr_t(w=2)^{W_2} = 1/2$*. In the world* W_4*,* $w = 2$ *has a degree of responsibility* $dr_t(w=2)^{W_4} = 1$*, because it is a counterfactual cause.*

```
function ComputeCauses(λ_a^{Q,D})
  for each λ_a^{Q,W} in λ_a^{Q,D} do
    for each minterm C in λ_a^{Q,W} do
      Causes_W = Causes_W ∪ < V_i = v_i; P(V_i = v_i) >
    end for
    Causes = Causes ∪ Causes_W
  end for
end function
function ComputeRespons(Causes)
  for each Distinct cause C in Causes_W do
    K = 0
    for each minterm M in λ_a^{Q,W} do
      if C ∉ M then
        K = K+1
      end if
    end for
    dr(C) = 1/1 + k
    Pr(C) = Pr(W)
    drP(C) = drP(C) + (drP(C) × Pr(C))
  end for
end function
function ComputeCauses(Causes with dr and Pr)
  for each uncertain attribute X do
    for each cause C in Causes_W: C ∈ X_T^W do
      dB(X) = Σ_{W∈W_t} Π_{t∈T_{V_i^W}} P(V_i = v_i) dr_t(V_i = v_i)
    end for
  end for
end function
function OutputExplanations(Causes with dR and Pr,
Attributes with dB)
    Output Causes ordered by dR × Pr
    Output Attributes ordered by dB
end function
```

W	P						
W1	0.0022	(x = 1,0.4)	(u = 2,0.3)	(y = 2,0.25)	(v = 2,0.25)	(z = 4,0.3)	(w = 2,1)
W2	0.022			(y = 2,0.25)	(v = 2,0.25)	(z = 4,0.3)	(w = 2,1)
W3	0.0033	(x = 2,0.6)	(u = 2,0.3)			(z = 4,0.3)	(w = 2,1)
W4	0.055					(z = 3,0.7)	(w = 2,1)

Fig. 4. Causality matrix.

Now for computing probabilistic responsibility, we have all information in hand, so the probabilistic responsibilities of these causes are computed as follows:

$$drP_t(w=2) = dr_t(w=2)^{W_1} \times P(W_1) + dr_t(w=2)^{W_2} \times P(W_2) + dr_t(w=2)^{W_3} \times P(W_3) + dr_t(w=2)^{W_4} \times P(W_4)$$
$$= (1/3) \times 0.0022 + (1/2) \times 0.022 + (1/2) \times 0.0033 + (1) \times 0.055 = 0.0683$$

$$drP_t(z=4) = dr_t(z=4)^{W_1} \times P(W_1) + dr_t(z=4)^{W_2} \times P(W_2) + dr_t(z=4)^{W_3} \times P(W_3)$$
$$= (1/3) \times 0.0022 + (1/2) \times 0.022 + (1/2) \times 0.0033 = 0.013$$

$$drP_t(z=3) = dr_t(z=3)^{W_4} \times P(W_4) = 1 \times 0.055 = 0.055$$

We notice that $drP_1(z=3) > drP_1(z=4)$ *although* $z=4$ *is included in more worlds than* $z=3$ (W_1, W_2, W_3). *However, since* $z=3$ *is a counterfactual cause in a world with a high probability, that makes it more responsible cause. It is evident here that the most responsible cause is* $w=2$, *since it is included in all worlds.*

Now let us compute the blame for each uncertain attribute SSN and M. We recall that variables x,y and z are defined under the attribute SSN, whereas u,v and u are defined under the variable M.

$$db(SSN) = [(dr_t(x=1)^{W_1} \times P(x=1)) \times (dr_t(y=3)^{W_1} \times P(y=3)) \times (dr_t(z=4)^{W_1} \times P(z=4))] + [(dr_t(y=1)^{W_2} \times P(y=1)) \times (dr_t(z=4)^{W_2} \times P(z=4))] + [(dr_t(x=2)^{W_3} \times P(x=2)) \times (dr_t(z=4)^{W_3} \times P(z=4))] + [(dr_t(z=4)^{W_4} \times P(z=4))]$$
$$= (0.4 \times 0.3 \times 0.3 \times 1/3) + (0.7 \times 0.3 \times 1/2) + (0.6 \times 0.3 \times 1/2) + 0.7 \times 1 = 0.907$$

$$db(M) = [(dr_t(u=2)^{W_1} \times P(u=2)) \times (dr_t(v=2)^{W_1} \times P(v=2)) \times (dr_t(w=2)^{W_1} \times P(w=4))] + [(dr_t(v=2)^{W_2} \times P(v=2)) \times (dr_t(w=2)^{W_2} \times P(w=2))] + [(dr_t(u=2)^{W_3} \times P(u=2)) \times (dr_t(w=4)^{W_3} \times P(w=2))] + [(dr_t(w=2)^{W_4} \times P(w=2))]$$
$$= 0.3 \times 0.25 \times 1 \times 1/3 + 0.25 \times 1 \times 1/2 + 1 \times 1/2 + 1 \times 1 = 1.314$$

We see here that $db(M) > db(SSN)$, *that is M contributes the most for the probability of the output result of SSNs for our query. One evident difference between the two attributes is the certain tuple with the valuation* $w=1$ *and responsibility one in* W_4. *This shows clearly that the attribute having the most blame is the one consisting of more responsible and certain tuples.*

We should mention that this measure informs us clearly about which attribute contributes more to the uncertainty that impacts the output probability. Let us suppose the case where SSNs are usually certain, and just under some cases the probability is distributed over two possible values at most, and for M, we suppose that along all forms the value of M is never certain and might take all 4 possible values. Considering a case like this would result evidently in $db(SSN) > db(M)$, *i.e, SSN contributes the most for the output probability. Actually, knowing such information will have a high importance through enabling us to address the source of this uncertainty in the future.*

V. Experimental Evaluation

We implemented the above algorithm in C#. We tested our method in Windows 10 with i7 CPU and 8 GB memory. To our knowledge, there is no benchmark of probabilistic databases available for performing experiments, therefore, we have dealt with this issue through two main experiments on synthetic and real data respectively. In the first experiment, we created a probabilistic database that concerns the census scenario by creating a list of random forms, and then we tried to compute causes and responsibilities. This experiment is meant mainly to measure the performance of our method in terms of execution time. The second experiment is based on a real data set extracted from the IMDB database [24]. In this experiment we will show the usefulness of causality, responsibility and blame for explaining probabilistic query answers.

A. Synthetic Census Data Set

We make two basic experiments on the synthetic census data set. The first is done by fixing the number of worlds by 500 and varying the number of forms. We compute the time execution in seconds for different numbers of forms, from 5000 to 30000 forms.

The result of this experiment is depicted in Fig. 5. For 5000 forms, the time execution of our algorithm is estimated by 0.79 seconds, and 1.36 seconds for 10000, and continues increasing, until it reaches 10.57 seconds for 30000 forms.

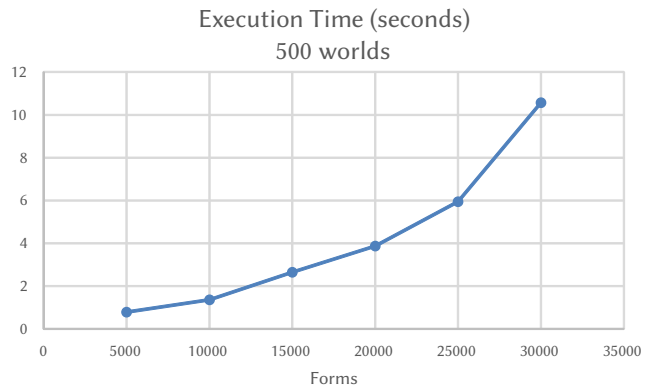

Fig. 5. Execution time with respect to number of forms.

We perform another experiment now by fixing the number of forms by 50000, and varying the number of worlds from 100 to 600 (see Fig. 6). For 100 worlds, the time taken by our algorithm is estimated by 1.36 seconds, and 3.71 for 200 *worlds* until it reaches 19.60 seconds for 600 worlds. As a result of these two experiments, we see here that the execution time is affected by both the number of forms and worlds in similar way, and the computation time is efficient even for large sizes. In general, we observe that computing explanations is linear in the size of the number of forms as well as as the number of worlds. In other words it is linear in the size of the lineage, which conforms with the results found by Kanagal and Deshpande [13], where the computation times are also very similar.

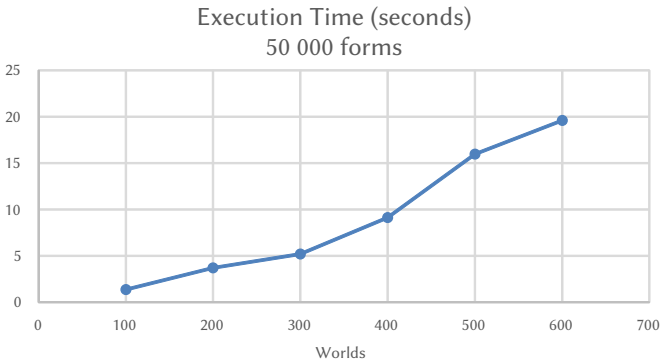

Fig. 6. Execution time with respect to number forms given 100 worlds.

Now we are going to present the results of the number of probabilistic responsibility classes based on the same previous experiments. So, we first fix the number of worlds by 500 and change the number of forms, and then fix the number of forms by 50000 and change the number of worlds. The results are depicted in Fig. 7 and Fig. 8 respectively. As we see in Fig. 7, it is evident that the number of causes and their probabilistic responsibilities increases in function of the number of forms. The second observation, which is more important, is that we have a small number of classes comparing to the number of causes. For instance, for 5000 forms, we have 2506002 causes and just 455 classes of probabilistic responsibilities, and given 50 000 forms, we have over 17 million (17536002) causes with 501 classes only. We can explain this result by having a large set of causes that share the same probabilistic responsibility value. Regrouping together all the causes that share the same probabilistic responsibility would be very useful. However, we notice that the number of classes is close from a value to another, where is still constant starting from 20000 forms. This is can be explained as the following: as long as the number of causes increases, we will have equal probabilistic responsibilities values, and thus no new classes are created. We should mention here that the number of causes does not refer to distinct causes, but rather to the number of causes counted over all the worlds, which means that the causes counted in a world are counted again in another world, which is required, since we are interested in computing probabilistic responsibilities at each world.

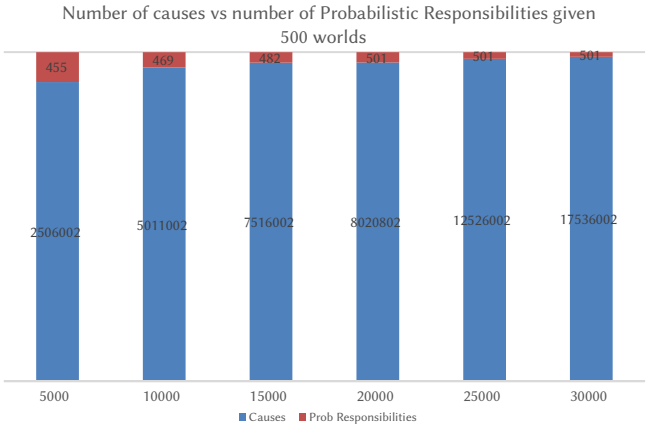

Fig. 7. Causes and probabilistic responsibility in function of forms number.

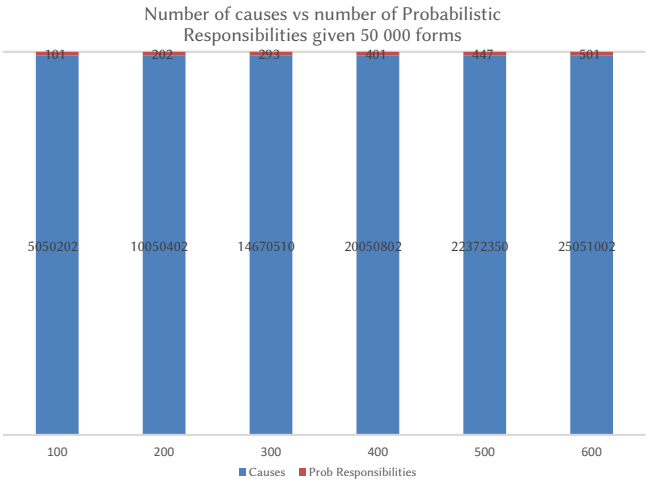

Fig. 8. Causes and probabilistic responsibility in function of worlds number.

Concerning the second experiment, we see that the results depicted in Fig. 8 are similar to the previous results. It is evident that as the number of worlds increases, we obtain a large set of causes as well as probabilistic responsibility classes. We notice that the maximum value of classes number, which is 501 is reached at 600 worlds, which refers exactly to the same maximum value found in the previous experiment.

Now we are going to present a final experiment that aims to show the impact of fixing the number of probabilistic responsibility classes on speeding up the execution time. Fixing the maximum value for probabilistic responsibilities is desirable by the user in order to focus only on most responsible causes and omitting the causes of low importance, which is one of the objectives of our work. To better show the impact of fixing the number of classes, we depict in Fig. 9 the execution time for top 100 classes together with the results of the first experiment. As we see in the results, the execution time is remarkably reduced. For instance for 5000 forms, the time is reduced from 0.79 to 0.24, from 1.36 to 0.38 for 10000 forms, and finally for 30000 forms, the time is reduced from 10.57 seconds to 1.14 seconds. So, relying on most responsible causes only allows to focus on important causes, and thus it helps to deliver good results in term of execution time, since we are omitting a large set of causes that share the rest of probabilistic responsibility classes, which are estimated for instance for 30000 forms by 400 classes.

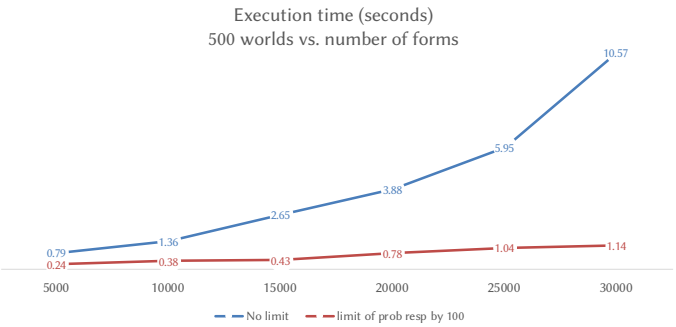

Fig. 9. Difference in execution time for 100 classes of probabilistic responsibility.

B. IMDB Data Set

This experiment is based on a prepared probabilistic database that has uncertain relations, where the sources of uncertainty are related to fuzzy object matches for titles, and the confidence in movies ratings [46]. The probabilistic IMDB database consists of two main relations. The schema of this database is presented in Fig. 10. While the first relation introduces uncertainty on the level of title, the second introduces uncertainty on the level of ratings, where a rating has a value in the range: 1 to 5.

```
Movies(movie_id int, title varchar,
year int, director varchar)
Ratings(movie_id int, cust_id int,
date varchar, rating int, confidence float)
```

Fig. 10. IMDB probabilistic database schema.

The movies relation consists of 1881 tuples, and the ratings relation consists of 10037 tuples. We aim to execute a query that returns the possible ratings for such a movie. All the queries required for returning this result are introduced in the database management system MayBMS [22]. Based on the possible worlds semantics, we must first repair the database to enforce constraints and make it consistent through the following queries:

```
create table ratings_1 as
select rating, movie_id
from (repair key movie_id in
ratings weight by confidence) r;
```

```
---------------------------------------
create table movies_1 as
select movie_id, title, director, year
from (repair key movie_id in movies) q;
```

Now, we estimate the exact confidence of distinct tuples through *conf()* function, such as: what are the possible ratings for each movie given all the instances of the probabilistic database. *Conf()* is computed as the sum of the probabilities of the instances (worlds) in which the tuple occurs as follows:

```
create table ratings_1_conf as
select rating, movie_id, conf()
from ratings_1
group by movie_id, rating;
 ---------------------------------
create table movies_1_conf as
select title, movie_id, director,
year, conf()
from movies_1
group by movie_id, title, director, year;
```

Now, we can join the two relations into a third one to obtain complete information using the following query:

```
create table ratings_movies as
select M.movie_id, M.title,
M.year, R.rating, R.conf
from movies_1_conf as M,
ratings_1_conf as R where
M.movie_id=R.movie_id
group by M.movie_id, M.title,
M.year, M.director, R.rating,
R.conf order by M.movie_id,
```

The resulting relation *ratings_movies* consists of 3292 tuples. Given this relation, we now want to get the possible ratings of such a movie. We choose *Jack* as a title for our query. The query as well as its results is given below:

```
select distinct rating, conf from (select *
from ratings_movies_conf where title
like '%Jack%') Q;
-------------------------------------
rating       | conf
-------------------------------------
1 | 0.12
2 | 0.02
2 | 0.2
2 | 0.3
3 | 0.42
3 | 0.52
3 | 0.56
4 | 0.06
4 | 0.12
4 | 0.32
5 | 0.12
5 | 0.24
```

As we see in the results, the user can get different confidence values for different ratings values, where all ratings values are present from 1 to 5. The user can have in mind different questions regarding these results, and he would be seeking for some explanations. Why I have rating 1 in my results, while I'm expecting at least 3? why 3 has the highest probabilities values, while I'm expecting the movie to have rating 5? is there a mismatch with other movies? so who is the director of this movie, and in which year ? etc. In other words, which tuples are responsible for such an outcome.

The tuples responsible for the presented output are 248 tuples, which can be returned by the following query:

```
$select * from ratings_movies_conf
where title like '\%Jack\%'$).
```

We see here that for a specific requested movie (Jack), we have 248 tuples that have contributed, which clearly shows that the evaluation of a lineage formula, and extracting the most responsible causes for a query answer in probabilistic databases is a challenging task. For each resulting rating value, we investigate the lineage information and the most responsible causes. For the value 1, there is one tuple that has contributed, which is a conjunction of two tuples from movies and ratings relations. These tuples are: *movies*(581, *BlackJack* : *Themovie,* 1993,*Unknown*) and *ratings*(581, 1), which are returned as an explanation for the rating 1. Both tuples have a degree of responsibility 1, since both tuples are counterfactual causes, where removing one tuple no longer yields a rating 1. However, the tuple *movies*(581, *BlackJack*: *Themovie,* 1993,*Unknown*) is the most responsible cause, since it is certain, thus it has a degree of probabilistic responsibility (*drP*) greater than the tuple *ratings*(581, 1), which is not certain. This example clearly shows the need for probabilistic responsibility, where classical responsibility is not enough for explaining probabilistic databases.

In contrast to rating 1, for the values 2, 3, and 4,we have 62 tuples contributing to the query answers, whereas for rating 5, we have 61 tuples, which results in 248 tuples in total. An example of results for rating 2, rating 3 and rating 4 are presented in Fig. 11. For all of these values, we see that we have exactly two tuples of the movies 508, and 581, whereas for the rest of 60 tuples, they are all related to the movie 172. This means that the movies 508 and 581 are certain tuples, whereas the movie 172 introduces a high level of uncertainty with different matches on titles. For instance, we have the same title *Jack* in different years (1913, 1916,...) and different titles matches (*Jack, Jacky, Jill Jacks Off, ...).* Now we want to compute the probabilistic responsibility for both movies and ratings tuples.

Since the movies 508 and 581 are certain, they appear in all the worlds of this probabilistic database, and since in every world we have 3 movies with the title *Jack*, the responsibility of these causes is 1/3, and thus for rating 2, the most responsible cause is the tuple: *movies(581, Black Jack: The movie, 1993, Unknown)*, since it has the highest degree of probabilistic responsibility (*drP*). For rating 3, the confidence seems high and close for every tuple, which means that it is the highly possible rating for all movies. Although, the Sixteen (60) movies with the *movie_id* = 172 seem to have the highest confidence values, every tuple can occur only once at each world, in contrast to the two previous ones (508 and 581), which are certain, and therefore the most responsible cause is still the same (*t:movies(581, Black Jack: The movie, 1993, Unknown)*). The results for rating 4 are very similar, however, the most responsible cause this time is *movies(508, Siant Jack, 1979, Bogdanovich Peter).* For rating 5, we have only two movies that are candidate to be causes, the movie 172 and 508, where 581 has no responsibility at all. The most responsible cause is *movies(508, Siant Jack, 1979, Bogdanovich Peter)*, since it is a certain tuple, and its probability of being ranked 5 is higher than the movie 172. To clarify more the importance of these results for explanation, let us consider the following scenario: when the user asks for the rating of *Jack*, he was anticipating at least rating 4, which makes him surprised getting ratings: 1, 2 and 3. From the previous results, he would know that the movie is looking for it might be *movies(508, Siant Jack, 1979, Bogdanovich Peter)*, since it has no possibility of being ranked 1, ranked 2 and 3 with low probabilities, and ranked 4 and 5 with high probabilities (most responsible).

	Query results for rating 2	Query results for rating 3	Query results for rating 4
1	508 | Saint Jack | 1979 | Bogdanovich Peter | 2 | 0.02	508 | Saint Jack | 1979 | Bogdanovich Peter | 3 | 0.42	508 | Saint Jack | 1979 | Bogdanovich Peter | 4 | 0.32
2	581 | Black Jack: The Movie | 1993 | Unknown | 2 | 0.3	581 | Black Jack: The Movie | 1993 | Unknown | 3 | 0.52	581 | Black Jack: The Movie | 1993 | Unknown | 4 | 0.06
3	172 | Jack | 1913 | Liabel Andre | 2 | 0.2	172 | Jack | 1913 | Liabel Andre | 3 | 0.56	172 | Jack | 1913 | Liabel Andre | 4 | 0.12
4	172 | Jack | 1916 | Borzage Frank | 2 | 0.2	172 | Jack | 1916 | Borzage Frank | 3 | 0.56	172 | Jack | 1916 | Borzage Frank | 4 | 0.12
5	172 | Jacky | 2000 | Hu Fow Pyng | 2 | 0.2	172 | Jacky | 2000 | Hu Fow Pyng | 3 | 0.56	172 | Jacky | 2000 | Hu Fow Pyng | 4 | 0.12
	.....	.....	.....
62	172 | Jill Jacks Off | 1993 | Constantinou, S.D | 2 | 0.2	172 | Jill Jacks Off | 1993 | Constantinou, S.D | 3 | 0.56	172 | Jill Jacks Off | 1993 | Constantinou, S.D | 4 | 0.12

Fig. 11. Query results for ratings 2–4.

Now concerning blame, we want to measure the degree of blame for each uncertain attribute, which are *title* and *rating*. Actually, the degrees of blame of the previous query are very close, however, for most values of ratings, the degree of blame of *title* is higher than the degree of blame for *rating* ($db(title) > db(rating)$), which means that it has more contribution for the estimated probabilities, and thus *rating* has the highest source of uncertainty. The results can be explained as the following: as we showed before, as much as the uncertain relation has certain tuples, as much as the degree of blame(dB) increases. Actually, it is the case here for the attribute *title* with the two movies 508 and 581, which are certain, however for the movie 172, we have a high level of uncertainty with sixty (60) possible titles match, which decreases significantly the degree of blame for this uncertain attribute. Though, the degree of blame for *rating* is low comparing to *title*, because all the movies concerned with this query have no certain rating values. Now, if we suppose that the movie 172 is certain as well, that will result absolutely in increasing significantly the degree of blame of *title*. We should mention that this kind of information is not just helpful for estimating the level of uncertainty for each uncertain attribute on the local level of a query answer, but also it can give us a better insight on the entire probabilistic database and how these attributes affect our queries results.

VI. Conclusion and Future Work

In this paper, we have shown how causality, responsibility and blame can be used in a complementary way in order to give useful quantitative explanations for query results in probabilistic databases. We presented how probabilistic responsibility measure could help us to identify which tuples have contributed the most for a query answer. In addition, we employed blame to identify among many uncertain attributes, which attribute has the most blame for such an outcome. This technique enables us to obtain a complete explanation framework. This technique has been proved to be useful in probabilistic database management systems that feature U-relational model like MayBMS. We delivered an algorithm for computing explanations, which has been implemented and tested on synthetic as well as real data sets.

While our technique addresses only conjunctive queries, among future works is the study of other types of complex queries, such as aggregation and top-k queries. Furthermore, this framework may benefit from employing sufficient lineage instead of complete lineage, which produces a smaller approximate lineage formula.

References

[1] Carnegie Mellon University, "Read the Web" Research Project Website Accessed: Oct. 19, 2022. [online]. available: http://rtw.ml.cmu.edu/rtw//.

[2] D. Suciu, D. Olteanu, C. Re, C. Koch, *Probabilistic Databases*. Morgan and Claypool Publishers, 2010.

[3] P. Bosc, O. Pivert, "Modeling and querying uncertain relational databases: A survey of approaches based on the possible worlds semantics," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 18, no. 5, pp. 565–603, 2010.

[4] N. Dalvi, D. Suciu, "Efficient query evaluation on probabilistic databases," *The International Journal on Very Large Data Bases*, vol. 16, no. 4, pp. 523–544, 2007.

[5] X. Dong, Y. Luming, "Study on consistent query answering in inconsistent databases," *Frontiers of Computer Science in China*, vol. 1, no. 4, pp. 493–501, 2007.

[6] M. Arenas, L. E. Bertossi, J. Chomicki, "Consistent query answers in inconsistent databases," in *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 1999, pp. 68–79.

[7] F. Parisi, J. Grant, "On repairing and querying inconsistent probabilistic spatio-temporal databases," *International Journal of Approximate Reasoning*, vol. 84, pp. 41–74, 2017.

[8] M. Calautti, L. Libkin, A. Pieris, "An operational approach to consistent query answering," in *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2018, pp. 239–251.

[9] X. Wang, X. L. Dong, A. Meliou, "Data x-ray: A diagnostic tool for data errors," in *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2015, pp. 1231–1245.

[10] C. Berkholz, M. Merz, "Probabilistic databases under updates: Boolean query evaluation and ranked enumeration," in *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2021, p. 402–415.

[11] L. Antova, T. Jansen, C. Koch, D. Olteanu, "Fast and simple relational processing of uncertain data," in *In Proc. 24th IEEE International Conference on Data Engineering*, 2008, pp. 983–992.

[12] G. V. den Broeck, D. Suciu, "Query processing on probabilistic data: A survey," *Foundations and Trends in Databases*, vol. 7, no. 4, pp. 197–341, 2015.

[13] B. Kanagal, J. Li, A. Deshpande, "Sensitivity analysis and explanations for robust query evaluation in probabilistic databases," in *ACM SIGMOD International Conference on Management of Data*, 2011, pp. 841–852.

[14] C. Re, D. Suciu, "Approximate lineage for probabilistic databases," *The International Journal on Very Large Data Bases*, vol. 1, no. 1, pp. 797–808, 2008.

[15] I. Ceylan, S. Borgwardt, T. Lukasiewicz, "Most probable explanations for probabilistic database queries," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017, pp. 950–956.

[16] J. Y. Halpern, J. Pearl, "Causes and explanations: A structural-model approach part i: Causes," in *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, 2001, pp. 194–202.

[17] J. Y. Halpern, J. Pearl, "Causes and explanations: A structural-model approach. part ii: Explanations," *British Journal for the Philosophy of Science*, vol. 56, no. 4, pp. 889–911, 2008.

[18] H. Chockler, J. Y. Halpern, "Responsibility and blame: a structural model approach," *Journal of Artificial Intelligence Research (JAIR)*, vol. 22, no. 1, pp. 93–115, 2004.

[19] A. Meliou, W. Gatterbauer, J. Y. Halpern, C. Koch, "Causality in databases," *IEEE Data Engineering Bulletin*, vol. 33, no. 3, pp. 59–67, 2010.

[20] L. Bertossi, B. Salimi, "Causes for query answers from databases: Datalog abduction, view-updates, and integrity constraints," *International Journal of Approximate Reasoning*, vol. 90, pp. 226–252, 2017.

[21] L. Antova, T. Jansen, C. Koch, D. Olteanu, "Fast and simple relational processing of uncertain data," in *IEEE 24th International Conference on*

Data Engineering, 2008, pp. 983–992.

[22] L. Antova, C. Koch, D. Olteanu, "Maybms:managing incomplete information with probabilistic world–set decompositions," in *In Proc. 23rd IEEE International Conference on Data Engineering*, 2007, pp. 1479–1480.

[23] D. Suciu, "Probabilistic databases for all," Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, 2020, p. 19–31.

[24] IMDb.com, Inc., IMDb Website. Accessed: Oct. 19, 2022. [Online]. Available: https://www.imdb.com//.

[25] A. Meliou, W. Gatterbauer, K. Moore, D. Suciu, "The complexity of causality and responsibility for query answers and non-answers," *The International Journal on Very Large Data Bases*, vol. 4, no. 1, pp. 34–45, 2010.

[26] T. J. Green, V. Tannen, "Models for incomplete and probabilistic information," in *Proceedings of the international conference on Current Trends in Database Technology*, 2006, pp. 278–296.

[27] T. J. Green, G. Karvounarakis, V. Tannen, "Provenance semirings," in *Proceedings of the 2007 ACM SIGMOD- SIGACT-SIGAI Symposium on Principles of Database Systems*, 2007, pp. 31–40.

[28] R. Diestelkämper, S. Lee, M. Herschel, B. Glavic, *To Not Miss the Forest for the Trees - A Holistic Approach for Explaining Missing Answers over Nested Data*, p. 405–417. 2021.

[29] B. Salimi, *Quer-Answer Causality in Dataabses And its Connections with Reverse Reasoning Tasks in Data And Knowledge Management*. PhD dissertation, Carleton University, 2015.

[30] L. Bertossi, B. Salimi, "From causes for database queries to repairs and model-based diagnosis and back," *Theory of Computing Systems*, vol. 61, no. 1, pp. 191–232, 2017.

[31] A. Meliou, A. Meliour, S. Nath, D. Suciu, "Tracing data errors with view-conditioned causality," in *Proceedings of the 2011 ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2011, pp. 505–516.

[32] X. Lian, L. Chen, "Causality and responsibility: probabilistic queries revisited in uncertain databases," in *Proceedings of the 22nd ACM international conference on Information and Knowledge Management*, 2013, pp. 349–358.

[33] K. Mu, "Responsibility for inconsistency," *International Journal of Approximate Reasoning*, vol. 61, pp. 43–60, 2015.

[34] Z. Miao, Q. Zeng, B. Glavic, S. Roy, "Going beyond provenance: Explaining query answers with pattern-based counterbalances," in *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, 2019, p. 485–502.

[35] L. Antova, C. Koch, D. Olteanu, "From complete to incomplete information and back," in *ACM SIGMOD International Conference on Management of Data*, 2007, pp. 713–724.

[36] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Ré, D. Suciu, "Mystiq: a system for finding more answers by using probabilities," in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, 2005, pp. 891–893.

[37] O. Benjelloun, A. D. Sarma, A. Halevy, J. Widom, "Databases with uncertainty and lineage," in *The International Journal on Very Large Data Bases*, 2006, pp. 953–964.

[38] O. Benjelloun, A. D. Sarma, C. Hayworth, J. Widom, "An introduction to uldbs and the trio system," *IEEE Data Engineering Bulletin*, vol. 29, no. 1, pp. 5–16, 2006.

[39] T. Imielin´ski, W. Lipski, "Incomplete information in relational databases," *Journal of the ACM (JACM)*, vol. 31, no. 4, pp. 761–791, 1984.

[40] P. Sen, A. Deshpande, L. Getoor, "Read-once functions and query evaluation in probabilistic databases," *The International Journal on Very Large Data Bases*, vol. 3, no. 1, pp. 1068–1079, 2010.

[41] D. Olteanu, J. Huang, "Using obdds for efficient query evaluation on probabilistic databases," in *2nd International Conference on Scalable Uncertainty Management*, 2008, pp. 109–121.

[42] A. Darwiche, P. Marquis, "A knowledge compilation map," *Journal of Artificial Intelligence Research*, vol. 17, no. 1, pp. 229–264, 2002.

[43] C. Re, N. Dalvi, D. Suciu, "Efficient top-k query evaluation on probabilistic data," in *2007 IEEE 23rd International Conference on Data Engineering*, 2007, pp. 108–122.

[44] C. Koch, D. Olteanu, "Conditioning probabilistic databases," *The International Journal on Very Large Data Bases*, vol. 1, no. 1, pp. 313–325, 2008.

[45] A. Meliou, W. Gatterbauer, K. Moore, D. Suciu, "Why so? or why no? functional causality for explaining query answers," Department of Computer Science and Engineering, University of Washington, Seattle, 2010.

[46] MayBMS Project, MayBMS - A Probabilistic Database Management System. Accessed: Oct. 19, 2022. [Online]. Available: http://maybms.sourceforge.net//.

Hichem Debbi

He received his Master's and PhD degrees in computer science from the University of M'sila, Algeria in 2011 and 2015 respectively. He is currently an assistant professor at the department computer science, University of M'sila. His research interests include but not limited to: causality, verification and explanation of probabilistic systems, debugging and analysing complex systems.