U

Uncertain Data Lineage

Sudeepa Roy
Department of Computer Science, Duke University, Durham, NC, USA

Synonyms

Provenance in probabilistic databases

Definition

Lineage, also called *Boolean provenance*, *event expression*, or *why-provenance*, is a form of *provenance or origin* of the answer(s) to a query executed on a database. Lineage is expressed as a Boolean formula with variables assigned to the tuples in the database, where joint usage of the tuples (by the database join operation) is captured by Boolean conjunction (AND, $\wedge$) and alternative usage (projection or union) by Boolean disjunction (OR, $\vee$). Uncertain data is typically expressed in the form of a *probabilistic database*, which is a compact representation of a probability distribution over a set of deterministic database instances (called *possible worlds*). When an input query is evaluated on such a probabilistic database, instead of a deterministic set of tuples representing the answer, the output is a distribution on possible answers for the possible worlds. The query evaluation problem on uncertain data aims to compute this output probability distribution efficiently. Lineages of the answers play a key role in understanding, expressing, and efficiently evaluating the probability distribution of query answers for uncertain data.

Historical Background

Several forms of **uncertain data**, either probabilistic or not, have been studied in the database literature starting in early 1980s, e.g., attribute-level uncertainty, world-set decomposition incomplete databases, probabilistic graphical model, possible world model, tree-based models (And/Xor Tree, WS-trees, decomposition trees), and tuple-independent and block-independent-disjoint probabilistic database models (see the books [2, 36] and the references therein). Pearl [33] introduced the taxonomy of intensional and extensional approaches as follows: the *extensional* (rule-based) approach treats uncertainty as a generalized truth value assigned to formulas and computes the uncertainty as a function of the uncertainties of its sub-formulas; the *intensional* (model-based) approach attaches uncertainty to subsets of *possible worlds*. According to this definition, the intensional approach is semantically clearer than extensional approach (due to potential *improper treatment of correlated sources of evidence* in the extensional approach) but computationally less efficient (due to the existence of many possible worlds). Later,

© Springer Science+Business Media LLC 2017
L. Liu, M.T. Özsu (eds.), *Encyclopedia of Database Systems*,
https://doi.org/10.1007/978-1-4899-7993-3_80759-1

Fuhr and Rölleke [18] (also [39]) defined an elaborate probabilistic data model, where each tuple of each relation (base, intermediate, or output) is accompanied by an *event expression*. Event expressions are manipulated by relational algebra (RA) operators, and the event expressions occurring in the base relations are associated with probabilistic weights. In this model, the tuple weights of the result of an event expression always conform to the underlying probabilistic model given by the intensional semantics, whereas for certain event expressions, the same result can be obtained by extensional semantics, assuming that the tuples in the base relations are independent. This *tuple-independent probabilistic database model* has been the most popular model in the study of the probabilistic databases so far, along with the concept of event expressions that are the most common notion of lineages used for probabilistic databases.

Lineage of an answer tuple output by a database query, in general, identifies the set of source tuples that contributed to the answer. Lineage was studied as a method to trace the sources of errors and debug anomalies in processed data (e.g., [10, 11]). However, one of the key usages of lineage has been in representing and evaluating uncertainty in the processed data. In fact, lineage was formally introduced in early 1980s in a seminal paper by Imielinski and Lipski [24] for representing query answers on incomplete data represented as *v-tables* (that may contain variables or *marked nulls*) and *c-tables* (v-tables where each tuple is annotated with a propositional formula and optional global conditions further restricting the possible worlds). Green and Tannen [20] discussed various relationships between incomplete databases and probabilistic databases and introduced the term *probabilistic c-table* or *pc-table* (for a c-table together with a finite probability space for each variable x that occurs in the c-table). Later, Green et al. [21] introduced the generic notion of *provenance semirings* and argued that both (i) conditions as propositional formulas restricting possible worlds on c-tables proposed by Imielinski and Lipski [24] and (ii) event expressions on event tables proposed by Fuhr and Rölleke [18] are special forms of provenance semirings. Lineage [5, 13] has been called by several names in the literature, e.g., event expressions [18], $PosBool[X]$ [21], *annotations* [16], or *Boolean provenance polynomials* $\mathbb{B}[X]$ (Green [19]); a concrete example of lineage in a probabilistic database is given in the next section.

Scientific Fundamentals

Lineage for query evaluation in tuple-independent probabilistic databases. In a tuple-independent probabilistic database, each input tuple has a certain probability belonging to a random instance of the database (called a possible world) independent of the other tuples. Figure 1a shows such an example database D (example from [34]) where each tuple t is annotated with a unique random *tuple variable* X_t and a probability value $p_t > 0$. For instance, for the tuple $t = R(a_1)$, $X_t = w_1$, and $p_t = 0.3$. The tuple t is present in an instance if and only if X_t is true (=1), $\Pr[X_t = 1] = p_t$, and X_t-s are independent random events. The probability of a possible world W can be computed by computing the probability that exactly the tuples in W are present and the other tuples are absent. Given a query Q, the probability of an answer tuple s belonging to $Q(D)$ is the sum of the probabilities of the possible worlds W of D such that $Q(W)$ contains s. Without loss of generality, it can be assumed that Q is a Boolean query, and therefore, the goal is to compute the probability that $Q(D)$ evaluates to true.

The number of possible worlds is exponential in the number of tuples in the input probabilistic database, so techniques have been developed to compute the distributions of the answers directly from the input representation of the probabilistic database D using event expressions [18] or lineages. The idea is to use conjunction ($\wedge$) for joint usage (joins) and disjunction ($\vee$) for alternative usage (unions or projections) on lineages of intermediate relations, starting with the tuple variables of the base relations. The resulting lineage is independent of the query plan used to evaluate

a

$R =$

a_1	w_1	(0.3)
b_1	w_2	(0.4)
a_2	w_3	(0.6)

$S =$

a_1	c_1	v_1	(0.1)
b_1	c_1	v_2	(0.5)
a_2	c_2	v_3	(0.2)
a_2	d_2	v_4	(0.1)

$T =$

c_1	u_1	(0.7)
c_2	u_2	(0.8)
d_2	u_3	(0.4)

b

$$Q() : -R(x), S(x,y), T(y)$$

c

$$\Phi_{Q,D} = w_1v_1u_1 + w_2v_2u_1 + w_3v_3u_2 + w_3v_4u_3$$

Uncertain Data Lineage, Fig. 1 A probabilistic database D, a Boolean query Q, and the lineage $\Phi_{Q,D}$

the answers [21] (i.e., different query plans will result in equivalent lineages) and is polynomial in size in data complexity (i.e., when Q is fixed and D variable). An example Boolean query Q and the corresponding lineage $\Phi_{Q,D}$ on the given database D is given in Fig. 1 (for the sake of simplicity, $+$ is used for OR ($\vee$), and AND ($\wedge$) is omitted in the Boolean formulas). It can be verified that the query Q will evaluate to true on the input database D if and only if $\Phi_{Q,D}$ is true, i.e., $\Pr[Q(D) = 1] = \Pr[\Phi_{Q,D} = 1]$, given the distributions of the independent random variables X_t-s.

Intensional and extensional methods. The event expression method, which has the same semantic as computation using explicit possible worlds, was called *intensional semantic* by Fuhr and Rölleke [18]. They also observed that the computation of answer probabilities may take exponentially many steps in general using the inclusion-exclusion principle. This is not surprising as the problem of query evaluation on probabilistic databases is #P-hard, even for very simple conjunctive queries (CQs) on tuple-independent probabilistic databases such as the one in Fig. 1b [14]. Fuhr and Rölleke, however, also observed that certain lineage expressions are amenable to probability computation in polynomial time by *extensional semantic* or *simple evaluation*, yielding the same result as in the intensional semantic. Extensional semantic has been studied thoroughly by a series of seminal papers by Dalvi and Suciu that eventually led to a dichotomy result [13] for union of conjunctive queries (UCQs): for any UCQ q, either a *safe query plan* exists that can compute the output probability in polynomial time for *any* input database (then q is a *safe query*) or computation of the probability for q is #P-hard (then q is an *unsafe query*). This article focuses on lineages for uncertain data and does not discuss the dichotomy result further.

Model counting on lineages for query evaluation. Intensional query evaluation on probabilistic databases reduces to *weighted-model counting* on the lineage expressed as a Boolean (propositional) formula. *Model counting* is the problem of computing the number, $\#\Phi$, of satisfying assignments of a Boolean formula Φ. *Weighted model counting* is the same as the probability computation problem on independent random variables, i.e., computing the probability of Φ being true given probabilities of the independent random variables in Φ. Model counting is #P-hard in general (even for formulas where satisfiability is easy to check (Valiant [37])). However, there are classes of propositional formulas that allow (weighted) model counting in polynomial time, for instance, those that can be efficiently compiled into certain *knowledge compilation forms* that are discussed later.

Lifted and grounded inference for model counting on lineages. The intensional approach of first computing the lineage and then evaluating the probability of the answers by weighted model counting is also called *grounded inference*. For example, given the Boolean query Q in Fig. 1b represented as a *first-order formula*: $\exists x, y\ R(x) \wedge S(x, y) \wedge T(y)$, and the domain of variables x, y as $\{1, 2, \cdots, n\}$ in a probabilistic database D, the lineage will be the *propositional formula* $\Phi_{Q,D} = \bigvee_{i,j \in [1,n]} R(i) \wedge S(i, j) \wedge T(j)$. The alternative to the grounded inference approach is the extensional query evaluation discussed earlier, also called *lifted inference* in the statistical relational model literature. There is no consensus on the definition of the lifted inference approach, except the intuitive idea that it should exploit the fixed query structure that is often lost in the grounded methods and should not require explicit construction of the lineage expression. Early definitions of lifted inference even required that the entire inference should be done using the first-order formula (query), without any grounding at all, in constant time in the size of the domain of the database. Such a restricted model called *symmetric probabilistic databases*, where probabilistic weights are attached to relations, has been studied by Beame et al. [6].

Knowledge compilation forms and their applications in weighted model counting. One approach to probability computation on a Boolean formula Φ is by compiling Φ into a *knowledge compilation form* K_Φ that allows efficient model counting. This approach requires that: (1) there exists a K_Φ of size polynomial in the size of Φ, and the conversion from Φ to K_Φ can be done in polynomial time, and (2) given K_Φ, the probability computation of Φ can be done in polynomial time in the size of K_Φ.

Compilation of Boolean formulas (in general, and for lineages in probabilistic databases) into the following knowledge compilation forms has been extensively studied in the literature. All of these forms allow computation of probability of the underlying propositional formula in time *linear* in the size of K_Φ, thus satisfying the second requirement above.

(a) **Read-once formulas**: A Boolean formula in which each variable occurs exactly once is in *read-once form* (Newman [31]) and the formulas that have an equivalent read-once form are called *read-once* [25, 34, 35]. For instance, $xy + yz + uw$ is not in read-once form (two occurrences of y), but it is read-once, since it is equivalent to $y(x + z) + uw$. If the participating variables in a formula in read-once form are mutually independent, the probability of the formula can be computed by a bottom-up pass on the corresponding *read-once tree* (see Fig. 2a) by repeatedly using $\Pr[X \wedge Y] = \Pr[X] \times \Pr[Y]$ and $\Pr[X \vee Y] = 1 - (1 - \Pr[X]) \times (1 - \Pr[Y])$. Of course, not all Boolean formulas are read-once (e.g., $xy + yz + zx$ or $xy + yz + zu$); a characterization is given by Gurvich [22] using the *co-occurrence graphs* of the formulas [35].

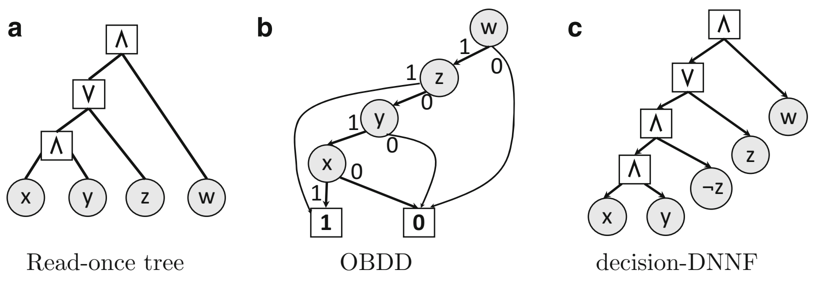
(b) **OBDDs and FBDDs**. Ordered binary decision diagrams (OBDDs) (Bryant [9]) are special kinds of *binary decision diagrams (BDDs)* (Akers [3]) or *branching programs* (Masek [29]) and represent a Boolean function using a directed acyclic graph (see Fig. 2b). Each internal node in this graph is a *decision* node, and there are two sink nodes labeled 0 and 1 (for false and true output values). A decision node queries a Boolean variable x in the formula and has 2 out edges labeled 0 and 1 denoting the value of the variable. Given an assignment of the Boolean variables, the value of the function is the label of the unique sink node reached from the root by following that assignment. Further, OBDDs require that each input variable is queried at most once on each source-sink path and the order of the variables in the decision nodes is the same along any path from the root to a sink. *Free binary decision diagrams (FBDDs)* (Wegener [38]), also known as *read-once branching programs (ROBPs)*, is a generalization of OBDDs where each variable is queried once but does not have to be queried in the same order along different paths.

(c) **Decision-DNNFs and d-DNNFs**. *Deterministic decomposable NNFs* (*d-DNNFs*) are restricted Boolean circuits in *negation normal form (NNF)* (circuits with AND ($\wedge$) and OR ($\vee$) gates with negations pushed to the input variables using De Morgan's laws). In a d-DNNF, the AND-nodes are decomposable (the sub-circuits are defined on disjoint set of variables), and the OR-nodes are deterministic (the sub-circuits never simultaneously evaluate to true); therefore, the probabilities of the sub-formulas at AND and OR nodes are, respectively, product and sum of the probabilities of their children. Decision-DNNFs (Huang and Darwiche [23]) are restricted d-DNNFs that ensure determinism by having a single variable x that evaluates to 1 on one branch of an OR-node and 0 on the other, $(x \land A) \lor (\neg x \land B)$, which is equivalent to having decision nodes as in FBDDs.

Lineages and knowledge compilation forms in probabilistic databases. Fuhr and Rölleke ([18], Thm. 4.5) stated that probabilities can be computed by "simple evaluation" (i.e., by the extensional method) if and only if the lineages computed intensionally are in read-once form. Moreover, the *safe plans* of [14] are such that if the lineages are computed with the plan, they will be in read-once form. Olteanu and Huang showed that if a conjunctive query without self-join Q is safe, then for any probabilistic database D, the lineage of any tuple in $Q(D)$ is read-once. Jha and Suciu [25] showed that (i) for CQs without self-joins, four classes of knowledge compilation forms (read-once, OBDDs, FBDDs, and d-DNNFs) collapse to the class of *hierarchical queries* [14] (the subsets of relational atoms for any pair of variables are either disjoint or one is contained in the other); (ii) however, for UCQs, these four classes form a strict hierarchy. Sen et al. [35] and Roy et al. [34] took an alternative approach and studied algorithms to decide the read-once property of lineages given *both the query and an input probabilistic database*. This approach allows efficient probability computation on read-once lineages of *unsafe* queries that are #P-hard in general (e.g., the query in Fig. 1b is unsafe, but its lineage $\Phi_{Q,D}$ in Fig. 1c on the input database D is read-once, with the equivalent read-once form: $(w_1 v_1 + w_2 v_2)u_1 + w_3(v_3 u_2 + v_4 u_3)$). Using the same instance-based approach, Amarilli et al. [4] studied algorithms and lower bounds for database instances with *bounded treewidths* (e.g., the lineage of monadic second-order queries on bounded-treewidth instances can be represented as bounded-treewidth circuits, polynomial-size OBDDs, and linear size d-DNNFs). Proving exponential lower bounds on the sizes of FBDDs and decision-DNNFs for a class of safe queries (with a polynomial time extensional evaluation technique), Beame et al. [5] argued that lifted inference or extensional evaluation is strictly more powerful than the grounded inference approach used in the state-of-the-art model counters that have to construct decision-DNNFs implicitly or explicitly.

Approximation. Lineages in probabilistic databases are also useful for approximate query evaluation. For very simple unsafe queries like the one in Fig. 1b, the query evaluation on probabilistic databases is #P-hard. However, for any positive query (UCQ), an approximation count with arbitrarily low error can be obtained (called fully polynomial randomized approximation scheme or *FPRAS*). The idea is to expand the propositional formula expressing the lineage into disjunctive normal form (DNF), which is polynomial in size for UCQs (data complexity), and then apply the Monte Carlo algorithm by Karp and Luby [27] for approximate DNF counting. Although the notion of lineages naturally extends to queries with the negation operator [18], the DNF-counting method does not apply to non-monotone queries since the DNF may be exponential in the size of the database. The complexity of query evaluation using lineages for queries with negation was studied by Khanna et al. [28], who showed that even the difference of two safe conjunctive queries can be #P-hard and, unlike positive queries without negation, inapproximable in general. Fink and Olteanu [15] investigated another concept of approximating query answers by obtaining Boolean formulas in read-once forms, that can provide upper or lower bounds of probabilities for the actual lineages of a given CQ without self-joins.

Lineages for other classes of queries. Fink and Olteanu [16] studied dichotomy for CQs with negation but no self-joins and for quantified queries expressing set division/inclusion/equivalence/incomparability, using compilation of the lineages into OBDDs. Fink et al. [17] studied positive RA queries with aggregates and proposed a representation system for probabilistic data called *pvc-tables* that can support aggregates as tuple annotations or lineage. Kanagal and Deshpande [26] studied probability computation using lineages when the tuples in the input probabilistic database are allowed to have arbitrary correlations. On such databases, even evaluation of read-once formulas was shown to be #P-hard,



Uncertain Data Lineage, Fig. 2 Various knowledge compilation forms for a simple read-once formula $(xy + z)w$. (**a**) Read-once tree. (**b**) OBDD. (**c**) decision-DNNF

and a characterization of the complexity was provided using a parameter called *lwidth* (similar to treewidth).

Key Applications

Lineages for uncertain data have been thoroughly studied for understanding query answering in probabilistic databases and have been applied in several systems for uncertain data. Benjelloun et al. [7] introduced the notion of *ULDBs (uncertain-lineage databases)* as a model of database systems that can support data in an extended relational model, an SQL-based query language, uncertainty of the data, and data lineage, all at the same time. The ULDB model was implemented in the Trio system [7] developed at Stanford, where each tuple has a bag of alternatives, and at most one of these alternatives can exist in a possible world (these are called *disjoint-independent* or *block-independent-disjoint probabilistic databases* [12]). Olteanu and Van Schaik proposed *ENTFrame* [32], a framework for processing probabilistic data using *event expressions* (a form of lineage) that can encode correlations, trace the computation of user programs (in a fragment of Python), and handle uncertainty in the program variables. Lineages have also been used for other applications on uncertain data, e.g., for *query containment* (Afrati and Vasilakopoulos [1]), *reasoning in uncertain RDF knowledge bases* (Meiser et al. [30]), and privacy and security in information integration activities (Blaustein et al. [8]).

Cross-References

▸ Data Provenance
▸ Lineage
▸ Probabilistic Databases
▸ Provenance
▸ Uncertain Databases

Recommended Reading

1. Afrati FN, Vasilakopoulos A. Query containment for databases with uncertainty and lineage. In: MUD; 2010. p. 67–81.
2. Aggarwal CC. Managing and mining uncertain data. New York: Springer Publishing Company, Incorporated; 2009.
3. Akers SB. Binary decision diagrams. IEEE Trans. Comput. 1978;27(6):509–16.
4. Amarilli A, Bourhis P, Senellart P. Tractable lineages on treelike instances: limits and extensions. In: PODS; 2016. p. 355–370.
5. Beame P, Li J, Roy S, Suciu D. Exact model counting of query expressions: limitations of propositional methods. ACM TODS. 2017;42(1):1:1–1:46.
6. Beame P, Van den Broeck G, Gribkoff E, Suciu D. Symmetric weighted first-order model counting. In: PODS; 2015. p. 313–28.
7. Benjelloun O, Sarma AD, Hayworth C, Widom J. An introduction to ULDBs and the Trio system. IEEE Data Eng. Bull. 2006;29(1):5–16.
8. Blaustein BT, Seligman L, Morse M, Allen MD, Rosenthal A. PLUS: Synthesizing privacy, lineage, uncertainty and security. In: ICDE workshops; 2008; p. 242–5.
9. Bryant RE. Graph-based algorithms for Boolean function manipulation. IEEE Trans. Comput. 1986;35(8):677–91.
10. Buneman P, Khanna S, Tan WC. Why and where: a characterization of data provenance. In: ICDT; 2001. p. 316–30.

11. Cui Y, Widom J, Wiener JL. Tracing the lineage of view data in a warehousing environment. ACM TODS. 2000;25(2):179–227.
12. Dalvi, N, Suciu, D. Management of probabilistic data: foundations and challenges. In: PODS; 2007. p. 1–12.
13. Dalvi N, Suciu D. The dichotomy of probabilistic inference for unions of conjunctive queries. J. ACM. 2013;59(6):30:1–87.
14. Dalvi NN, Suciu D. Efficient query evaluation on probabilistic databases. In: VLDB; 2004. p. 864–75.
15. Fink R, Olteanu D. On the optimal approximation of queries using tractable propositional languages. In: ICDT; 2011. p. 174–185.
16. Fink R, Olteanu D. Dichotomies for queries with negation in probabilistic databases. ACM TODS. 2016;41(1):4.
17. Fink R, Han L, Olteanu D. Aggregation in probabilistic databases via knowledge compilation. PVLDB. 2012;5(5):490–501.
18. Fuhr N, Rölleke T. A probabilistic relational algebra for the integration of information retrieval and database systems. ACM Trans. Inf. Syst. 1997;15(1):32–66.
19. Green TJ. Containment of conjunctive queries on annotated relations. In: ICDT; 2009. p. 296–309.
20. Green TJ, Tannen V. Models for incomplete and probabilistic information. IEEE Data Eng. Bull. 2006;29(1):17–24.
21. Green TJ, Karvounarakis G, Tannen V. Provenance semirings. In: PODS; 2007. p. 31–40.
22. Gurvich VA. Criteria for repetition-freeness of functions in the algebra of logic. Soviet Math. Dokl. 1991;43(3):721–6.
23. Huang J, Darwiche A. The language of search. JAIR. 2007;29:191–219.
24. Imielinski T, Lipski W Jr. Incomplete information in relational databases. J. ACM. 1984;31(4):761–91.
25. Jha AK, Suciu D. Knowledge compilation meets database theory: compiling queries to decision diagrams. In: ICDT; 2011. p. 162–73.
26. Kanagal B, Deshpande A. Lineage processing over correlated probabilistic databases. In: SIGMOD; 2010. p. 675–686.
27. Karp RM, Luby M. Monte-Carlo algorithms for enumeration and reliability problems. In: FOCS; 1983. p. 56–64.
28. Khanna S, Roy S, Tannen V. Queries with difference on probabilistic databases. PVLDB. 2011;4(11):1051–62.
29. Masek WJ. A fast algorithm for the string editing problem and decision graph complexity. Master's thesis, MIT; 1976.
30. Meiser T, Dylla M, Theobald M. Interactive reasoning in uncertain RDF knowledge bases. In: CIKM; 2011. p. 2557–2560.
31. Newman I. On read-once Boolean functions. In: Paterson MS, editor. Boolean function complexity. Cambridge/New York: Cambridge University Press; 1992. p. 25–34.
32. Olteanu D, van Schaik SJ. ENFrame: a framework for processing probabilistic data. ACM TODS. 2016;41(1):3:1–3:44.
33. Pearl J. Probabilistic reasoning in intelligent systems: networks of plausible inference. San Francisco: Morgan Kaufmann Publishers Inc; 1988.
34. Roy S, Perduca V, Tannen V. Faster query answering in probabilistic databases using read-once functions. In: ICDT; 2011. p. 232–43.
35. Sen P, Deshpande A, Getoor L. Read-once functions and query evaluation in probabilistic databases. PVLDB. 2010;3(1):1068–79.
36. Suciu D, Olteanu D, Christopher R, Koch C. Probabilistic databases. 1st ed. San Rafael: Morgan & Claypool Publishers; 2011.
37. Valiant LG. The complexity of enumeration and reliability problems. SIAM J. Comput. 1979;8(3):410–21.
38. Wegener I. Branching programs and binary decision diagrams: theory and applications. Philadelphia: SIAM; 2000. ISBN:0-89871-458-3.
39. Zimányi E. Query evaluation in probabilistic relational databases. Theor. Comput. Sci. 1997;171(1–2):179–219.