Models for Incomplete and Probabilistic Information

Todd J. Green
University of Pennsylvania
tjgreen@cis.upenn.edu

Val Tannen
University of Pennsylvania
val@cis.upenn.edu

1 Introduction

This is an abbreviated version of [13] where proofs and additional results are discussed (available also from `http://db.cis.upenn.edu`).

The representation of incomplete information in databases has been an important research topic for a long time, see the references in [12], in Ch.19 of [2], in [21], as well as the recent [22, 20, 19]. Moreover, this work is closely related to recently active research topics such as inconsistent databases and repairs [3], answering queries using views [1], and data exchange [9]. The classic reference on incomplete databases remains [14] with the fundamental concept of c-table and its restrictions to simpler tables with variables. The most important result of [14] is the query answering algorithm that defines an algebra on c-tables that corresponds exactly to the usual relational algebra ($\mathcal{RA}$). A recent paper [19] has defined a hierarchy of incomplete database models based on finite sets of choices and optional inclusion. One of our contributions consists of **comparisons** between the models [19] and the tables with variables from [14].

Two criteria have been provided for comparisons among all these models: [14, 19] discuss *closure* under relational algebra operations, while [19] also emphasizes *completeness*, specifically the ability to represent all finite incomplete databases. We point out that the latter is not appropriate for tables with variables over an infinite domain, and we contribute another criterion, **$\mathcal{RA}$-completeness**, that fully characterizes the expressive power of c-tables.

We also introduce a new idea for the study of models that are not complete. Namely, we consider combining existing models with queries in various fragments of relational algebra. We then ask how big these fragments need to be to obtain a combined model that is complete. We give a number of such **algebraic completion** results.

Early on, probabilistic models of databases were studied less intensively than incompleteness models, with some notable exceptions [5, 4, 18, 7]. Essential progress was made independently in three papers [10, 16, 23] that were published at about the same time. [10, 23] assume a model in which tuples are taken independently in a relation with given probabilities. [16] assumes a model with a separate distribution for each attribute in each tuple. All three papers attacked the problem of calculating the probability of tuples occurring in query answers. They solved the problem by developing more general models in which rows contain additional information (“event expressions”,“paths”,“traces”), and they noted the similarity with the conditions in c-tables.

We go beyond the problem of individual tuples in query answers by defining **closure** under a query language for probabilistic models. Then we develop a new model, **probabilistic c-tables** that adds *to the c-tables themselves* probability distributions for the values taken by their variables. Here is an example of such a representation that captures the set of instances in which Alice is taking a course that is Math with probability 0.3;

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Physics (0.3); or Chemistry (0.4), while Bob takes the same course as Alice, provided that course is Physics or Chemistry and Theo takes Math with probability 0.85:

Student	*Course*	*Condition*
Alice	x	
Bob	x	$x = \text{phys} \lor x = \text{chem}$
Theo	math	$t = 1$

$$x = \begin{cases} \text{math} & : 0.3 \\ \text{phys} & : 0.3 \\ \text{chem} & : 0.4 \end{cases} \qquad t = \begin{cases} 0: & 0.15 \\ 1: & 0.85 \end{cases}$$

The concept of probabilistic c-table allows us to solve the closure problem by using the same algebra on c-tables defined in [14].

We also give a **completeness** result by showing that probabilistic boolean c-tables (all variables are two-valued and can appear only in the conditions, not in the tuples) can represent *any* probabilistic database.

An important conceptual contribution is that we show that, at least for the models we consider, the probabilistic database models can be seen, as **probabilistic counterparts** of incomplete database models. In an incompleteness model a tuple or an attribute value in a tuple may or may not be in the database. In its probabilistic counterpart, these are seen as elementary events with an assigned probability. For example, the models used in [10, 16, 23] are probabilistic counterparts of the two simplest incompleteness models discussed in [19]. As another example, the model used in [7] can be seen as the probabilistic counterpart of an incompleteness model one in which tuples sharing the same key have an exclusive-or relationship.

A consequence of this observation is that, in particular, query answering for probabilistic c-tables will allow us to solve the problem of calculating probabilities about query answers for any model that can be defined as a probabilistic counterpart of the incompleteness models considered in [14, 19].

2 Incomplete Information and Representation Systems

Our starting point is suggested by the work surveyed in [12], in Ch. 19 of [2], and in [21]. A database that provides incomplete information consists of a *set of possible instances*. At one end of this spectrum we have the conventional single instances, which provide "complete information." At the other end we have the set of *all* allowable instances which provides "no information" at all, or "zero information."

We adopt the formalism of relational databases over a fixed countably infinite domain $\mathbb{D}$. We use the unnamed form of the relational algebra. To simplify the notation we will work with relational schemas that consist of a single relation name of arity n. Everything we say can be easily reformulated for arbitrary relational schemas. We shall need a notation for the set of *all* (conventional) instances of this schema, i.e., all the finite n-ary relations over $\mathbb{D}$ namely $\mathcal{N} := \{I \mid I \subseteq \mathbb{D}^n,\ I \text{ finite}\}$

Definition 1: An **incomplete(-information) database** (**i-database** for short), $\mathcal{I}$, is a set of conventional instances, i.e., a subset $\mathcal{I} \subseteq \mathcal{N}$.

The usual relational databases correspond to the cases when $\mathcal{I} = \{I\}$. The **no-information** or **zero-information database** consists of *all* the relations: $\mathcal{N}$.

Conventional relational instances are finite. However, because $\mathbb{D}$ is infinite incomplete databases are in general infinite. Hence the interest in finite, syntactical, representations for incomplete information.

Definition 2: A **representation system** consists of a set (usually a syntactically defined "language") whose elements we call tables, and a function *Mod* that associates to each table T an incomplete database *Mod*(T).

The classical reference [14] considers three representation systems: **Codd tables**, v**-tables**, and c**-tables**. v-tables are conventional instances in which variables can appear in addition to constants from $\mathbb{D}$. If T is a v-table then

$$\mathit{Mod}(T) := \{\nu(T) \mid \nu : \mathit{Var}(T) \to \mathbb{D} \text{ is a valuation for the variables of } T\}$$

Codd tables are v-tables in which all the variables are distinct. They correspond roughly to the current use of nulls in SQL, while v-tables model "labeled" or "marked" nulls. c-tables are v-tables in which each tuple is associated with a condition — a boolean combination of equalities involving variables and constants. We typically use the letter φ for conditions. The tuple condition is tested for each valuation ν and the tuple is discarded from $\nu(T)$ if the condition is not satisfied.

Example 1: Here is an example of a c-table.

$$S := \begin{array}{|ccc|c|}\hline 1 & 2 & x & \\ 3 & x & y & x = y \wedge z \neq 2 \\ z & 4 & 5 & x \neq 1 \vee x \neq y \\ \hline \end{array} \quad \mathit{Mod}(S) = \left\{ \begin{array}{|ccc|}\hline 1 & 2 & 1 \\ 3 & 1 & 1 \\ \hline \end{array}, \begin{array}{|ccc|}\hline 1 & 2 & 2 \\ 1 & 4 & 5 \\ \hline \end{array}, \ldots, \begin{array}{|ccc|}\hline 1 & 2 & 77 \\ 97 & 4 & 5 \\ \hline \end{array}, \ldots \right\}$$

Several other representation systems have been proposed in a recent paper [19]. We illustrate here three of them and we discuss several others later. A **?-table** is a conventional instance in which tuples are optionally labeled with "?," meaning that the tuple may be missing. An **or-set-table** looks like a conventional instance but or-set values [15, 17] are allowed. An or-set value $\langle 1, 2, 3\rangle$ signifies that exactly one of 1, 2, or 3 is the "actual" (but unknown) value. Clearly, the two ideas can be combined yielding another representation systems that we might (awkwardly) call **or-set-?-tables**.(In [19] these three systems are denoted by $\mathcal{R}_?$, $\mathcal{R}^A$ and $\mathcal{R}_?^A$.)

Example 2: Here is an example of an or-set-?-table.

$$T := \begin{array}{|ccc|c|}\hline 1 & 2 & \langle 1,2\rangle & \\ 3 & \langle 1,2\rangle & \langle 3,4\rangle & \\ \langle 4,5\rangle & 4 & 5 & ? \\ \hline \end{array} \quad \mathit{Mod}(T) = \left\{ \begin{array}{|ccc|}\hline 1 & 2 & 1 \\ 3 & 1 & 3 \\ 4 & 4 & 5 \\ \hline \end{array}, \begin{array}{|ccc|}\hline 1 & 2 & 1 \\ 3 & 1 & 3 \\ \hline \end{array}, \begin{array}{|ccc|}\hline 1 & 2 & 2 \\ 3 & 1 & 3 \\ 4 & 4 & 5 \\ \hline \end{array}, \ldots, \begin{array}{|ccc|}\hline 1 & 2 & 2 \\ 3 & 2 & 4 \\ \hline \end{array} \right\}$$

3 Completeness and Closure

"Completeness" of expressive power is the first obvious question to ask about representation systems. This brings up a fundamental difference between the representation systems of [14] and those of [19]. The presence of variables in a table T and the fact that $\mathbb{D}$ is infinite means that *Mod*(T) may be infinite. For the tables considered in [19], *Mod*(T) is always finite.

[19] defines completeness as the ability of a representation system to represent "all" possible i-databases. For the kind of tables considered in [19] the question makes sense. But in the case of the tables with variables in [14] this is hopeless for trivial reasons. Indeed, in such systems there are only countably many tables while there are uncountably many i-databases (the subsets of $\mathcal{N}$, which is infinite). We will discuss separately below *finite completeness* for systems that only represent finite database. Meanwhile, we will develop a different yardstick for the expressive power of tables with variables that range over an infinite domain.

c-tables and their restrictions (v-tables and Codd tables) have an inherent limitation: the cardinality of the instances in *Mod*(T) is at most the cardinality of T. For example, the zero-information database $\mathcal{N}$ *cannot* be represented with c-tables. It also follows that among the i-databases that are representable by c-tables the "minimal"-information ones are those consisting for some m of all instances of cardinality up to m (which are in fact representable by Codd tables with m rows). Among these, we make special use of the ones of cardinality 1:

$$\mathcal{Z}_k := \{\{t\} \mid t \in \mathbb{D}^k\}.$$

Hence, $\mathcal{Z}_k$ consists of *all* the one-tuple relations of arity k. Note that $\mathcal{Z}_k = \mathit{Mod}(Z_k)$ where Z_k is the Codd table consisting of a single row of k distinct variables.

Definition 3: An i-database $\mathcal{I}$ is $\mathcal{RA}$**-definable** if there exists a relational algebra query q such that $\mathcal{I} = q(\mathcal{Z}_k)$, where k is the arity of the input relation name in q.

Theorem 4: If $\mathcal{I}$ is an i-database representable by a c-table T, i.e., $\mathcal{I} = Mod(T)$, then $\mathcal{I}$ is $\mathcal{RA}$-definable.

Hence, c-tables are in some sense "no more powerful" than the relational algebra. But are they "as powerful"? This justifies the following:

Definition 5: A representation system is $\mathcal{RA}$-**complete** if it can represent any $\mathcal{RA}$-definable i-database.

Since Z_k is itself a c-table the following is an immediate corollary of the fundamental result of [14] (see Theorem 11 below). It also states that the converse of Theorem 4 holds.

Theorem 6: c-tables are $\mathcal{RA}$-complete.

We now turn to the kind of completeness considered in [19].

Definition 7: A representation system is **finitely complete** if it can represent any finite i-database.

The finite incompleteness of ?-tables, or-set-tables, or-set-?-tables and other systems is discussed in [19] where a finitely complete representation system called $\mathcal{R}^A_{\text{prop}}$ is also given (we do not discuss $\mathcal{R}^A_{\text{prop}}$further here). Is finite completeness a reasonable question for c-tables, v-tables, and Codd tables? In general, for such tables $Mod(T)$ is infinite (all that is needed is a tuple with at least one variable and with an infinitely satisfiable condition). To facilitate comparison with the systems in [19] we define *finite-domain* versions of tables with variables.

Definition 8: A **finite-domain** c-table (v-table, Codd table) consists of a c-table (v-table, Codd table) T together with a *finite* $\text{dom}(x) \subset \mathbb{D}$ for each variable x that occurs in T.

Note that finite-domain Codd tables are equivalent to or-set tables. Indeed, to obtain an or-set table from a Codd table, one can see $\text{dom}(x)$ as an or-set and substitute it for x in the table. Conversely, to obtain a Codd table from an or-set table, one can substitute a fresh variable x for each or-set and define $\text{dom}(x)$ as the contents of the or-set.

It is easy to see that finite-domain c-tables are finitely complete. In fact, this is true even for the fragment of finite-domain c-tables which we will call *boolean c-tables*, where the variables take only boolean values and are only allowed to appear in conditions (never as attribute values).

Theorem 9: Boolean c-tables are finitely complete (hence finite-domain c-tables are also finitely complete).

If we additionally restrict boolean c-tables to allow conditions to contain only true or a single variable which appears in no other condition, then we obtain a representation system which is equivalent to ?-tables.

Definition 10: A representation system is **closed** under a query language if for any query q and any table T there is a table T' that represents $q(Mod(T))$.

This definition is from [19]. In [2], a *strong* representation system is defined in the same way, with the significant addition that T' should be *computable* from T and q. It is not hard to show, using general recursion-theoretic principles, that there exist representation systems (even ones that only represent finite i-databases) which are closed as above but not strong in the sense of [2]. However, the concrete systems studied so far are either not closed, or if they are closed, as in the theorem below, then the proof provides also the algorithm required by the definition of strong systems. Hence, we see no need to insist upon the distinction.

Theorem 11 ([14]): c-tables are closed under the relational algebra. (The same proof works for finite-domain c-tables, and even boolean c-tables.)

4 Algebraic Completion

None of the incomplete representation systems we have seen so far is closed under the full relational algebra.

Proposition 12 ([14, 19]): Codd tables and v-tables are not closed under e.g. selection. Or-set tables and finite v-tables are also not closed under e.g. selection. ?-tables are not closed under e.g. join.

We have seen that "closing" minimal-information one-row Codd tables (see before Definition 5) $\{Z_1, Z_2, \ldots\}$, by relational algebra queries yields equivalence with the c-tables. In this spirit, we will investigate "how much" of the relational algebra would be needed to complete the other representation systems considered. We call this kind of result *algebraic completion*.

Definition 13: If $(\mathcal{T}, \mathit{Mod})$ is a representation system and $\mathcal{L}$ is a query language, then the representation system *obtained by closing $\mathcal{T}$ under $\mathcal{L}$* is the set of tables $\{(T, q) \mid T \in \mathcal{T}, q \in \mathcal{L}\}$ with the function $\mathit{Mod} : \mathcal{T} \times \mathcal{L} \to \mathcal{N}$ defined by $\mathit{Mod}(T, q) := q(\mathit{Mod}(T))$.

Theorem 14 ($\mathcal{RA}$-Completion): Closing Codd tables under $SPJU$ queries and closing v-tables under SP queries produces $\mathcal{RA}$-complete systems in both cases.

We give now a set of analogous completion results for the finite case.

Theorem 15 (Finite-Completion): Closing or-set-tables under PJ queries, closing finite v-tables under PJ or S^+P queries, and closing ?-tables under $\mathcal{RA}$ queries produces finitely complete systems.

5 Probabilistic Databases and Representation Systems

Finiteness assumption For the entire discussion of probabilistic database models we will assume that *the domain of values $\mathbb{D}$ is finite.* Infinite domains of values are certainly interesting in practice; for some examples see [16, 22, 19]. Moreover, in the case of incomplete databases we have seen that they allow for interesting distinctions.[1] However, finite probability spaces are much simpler than infinite ones and we will take advantage of this simplicity. We leave for future investigations the issues related to probabilistic databases over infinite domains.

We wish to model probabilistic information using a probability space whose possible outcomes are all the conventional instances. Recall that for simplicity we assume a schema consisting of just one relation of arity n. The finiteness of $\mathbb{D}$ implies that there are only finitely many instances, $I \subseteq \mathbb{D}^n$.

By **finite probability space** we mean a probability space (see e.g. [8]) $(\Omega, \mathcal{F}, \mathbb{P}[\ ])$ in which the set of outcomes Ω is *finite* and the σ-field of events $\mathcal{F}$ consists of *all* subsets of Ω. We shall use the equivalent formulation of pairs (Ω, p) where Ω is the finite set of outcomes and where the *outcome probability assignment* $p : \Omega \to [0, 1]$ satisfies $\sum_{\omega \in \Omega} p(\omega) = 1$. Indeed, we take $\mathbb{P}[A] = \sum_{\omega \in A} p(\omega)$.

Definition 16: A **probabilistic(-information) database** (or p-**database**) is a finite probability space whose outcomes are all the conventional instances, i.e., a pair $(\mathcal{N}, p)$ where $\sum_{I \in \mathcal{N}} p(I) = 1$.

Demanding the direct specification of such probabilistic databases is unrealistic because there are 2^N possible instances, where $N := |\mathbb{D}|^n$, and we would need that many (minus one) probability values. Thus, as in the case of incomplete databases we define **probabilistic representation systems** consisting of "probabilistic tables" (prob. tables for short) and a function *Mod* that associates to each prob. table T a probabilistic database $\mathit{Mod}(T)$. Similarly, we define **completeness** (finite completeness is the only kind we have in our setting).

[1]Note however that the results remain true if $\mathbb{D}$ is finite; we just require an infinite supply of *variables*.

To define closure under a query language we face the following problem. Given a probabilistic database $(\mathcal{N}, p)$ and a query q (with just one input relation name), how do we define the probability assignment for the instances in $q(\mathcal{N})$? It turns out that this is a common construction in probability theory: image spaces.

Definition 17: Let (Ω, p) be a finite probability space and let $f : \Omega \to \Omega'$ where Ω' is some finite set. The **image** of (Ω, p) under f is the finite probability space (Ω', p') where[2] $p'(\omega') := \sum_{f(\omega)=\omega'} p(\omega)$.

Again we consider as query languages the relational algebra and its sublanguages defined by subsets of operations.

Definition 18: A probabilistic representation system is **closed** under a query language if for any query q and any prob. table T there exists a prob. table T' that represents $q(\mathit{Mod}(T))$, the image space of $\mathit{Mod}(T)$ under q.

6 Probabilistic ?-Tables and Probabilistic Or-Set Tables

Probabilistic ?-tables (p-?-tables for short) are commonly used for probabilistic models of databases [23, 10, 11, 6] (they are called "independent tuple representation in [20]). Such tables are the probabilistic counterpart of ?-tables where each "?" is replaced by a probability value. Example 3 below shows such a table. The tuples not explicitly shown are assumed tagged with probability 0. Therefore, a p-?-table is a mapping that associates to each $t \in \mathbb{D}^n$ a probability value p_t.

To define the *Mod* function we use another common construction from probability theory: product spaces.

Definition 19: Let $(\Omega_1, p_1), \ldots, (\Omega_n, p_n)$ be finite probability spaces. Their **product** is the space $(\Omega_1 \times \cdots \times \Omega_n, p)$ where[3] $p(\omega_1, \ldots, \omega_n) := p_1(\omega_1) \cdots p_n(\omega_n)$.

Given a p-?-table $T := \{p_t \| t \in \mathbb{D}^n\}$ consider the finite probability space $B_t := (\{\mathsf{true}, \mathsf{false}\}, p)$ where $p(\mathsf{true}) := p_t$ and $p(\mathsf{false}) = 1 - p_t$ and then the product space $P := \prod_{t \in \mathbb{D}^n} B_t$.

We can think of its set of outcomes (abusing notation, we will call this set P also) as the set of functions from $\mathbb{D}^n$ to $\{\mathsf{true}, \mathsf{false}\}$, in other words, predicates on $\mathbb{D}^n$. There is an obvious function $f : P \to \mathcal{N}$ that associates to each predicate the set of tuples it maps to true and this gives us a p-database, namely the image of P under f, which we define to be $\mathit{Mod}(T)$.

We define now another simple probabilistic representation system called **probabilistic or-set-tables** (p-or-set-tables for short). These are the probabilistic counterpart of or-set-tables where the attribute values are, instead of or-sets, finite probability spaces whose outcomes are the values in the or-set. p-or-set-tables correspond to a simplified version of the ProbView model presented in [16], in which plain probability values are used instead of confidence intervals.

Example 3: A p-or-set-table S, and a p-?-table T.

$S :=$

1	$\langle 2 : 0.3, 3 : 0.7 \rangle$
4	5
$\langle 6 : 0.5, 7 : 0.5 \rangle$	$\langle 8 : 0.1, 9 : 0.9 \rangle$

$T :=$

1	2	0.4
3	4	0.3
5	6	1.0

A p-or-set-table determines an instance by choosing an outcome in each of the spaces that appear as attribute values, *independently*. Recall that or-set tables are equivalent to finite-domain Codd tables. Similarly, a p-or-set-table corresponds to a Codd table T plus for each variable x in T a finite probability space $\mathrm{dom}(x)$ whose

[2]It is easy to check that the $p'(\omega')$'s do actually add up to 1.

[3]Again, it is easy to check that the outcome probability assignments add up to 1.

outcomes are in $\mathbb{D}$. This yields a p-database, again by image space construction, as shown more generally for c-tables next in section 7.

Query answering The papers [10, 23, 16] have considered, independently, the problem of calculating the probability of tuples appearing in query answers. This does *not* mean that in general $q(\mathit{Mod}(T))$ can be represented by another tuple table when T is some p-?-table and $q \in \mathcal{RA}$ (neither does this hold for p-or-set-tables). This follows from Proposition 12. Indeed, if the probabilistic counterpart of an incompleteness representation system $\mathcal{T}$ is closed, then so is $\mathcal{T}$. Hence the lifting of the results in Proposition 12 and other similar results.

Each of the papers [10, 23, 16] recognizes the problem of query answering and solves it by developing a more general model in which rows contain additional information *similar in spirit* to the conditions that appear in c-tables (in fact [10]'s model is essentially what we call probabilistic boolean c-tables, see next section). We will show that we can actually use a probabilistic counterpart to c-tables themselves together with the algebra on c-tables given in [14] to achieve the same effect.

7 Probabilistic c-tables

Definition 20: A **probabilistic c-table** (**pc-tables** for short) consists of a c-table T together with a *finite probability space* $\mathrm{dom}(x)$ (whose outcomes are values in $\mathbb{D}$) for each variable x that occurs in T.

To get a probabilistic representation system consider the product space $V := \prod_{x\in \mathit{Var}(T)} \mathrm{dom}(x)$. The outcomes of this space are in fact the *valuations* for the c-table T! Hence we can define the function $g : V \to \mathcal{N}, g(\nu) := \nu(T)$ and then define $\mathit{Mod}(T)$ as the image of V under g.

Similarly, we can talk about boolean pc-tables, pv-tables and probabilistic Codd tables (the latter related to [16], see previous section). Moreover, the p-?-tables correspond to restricted boolean pc-tables, just like ?-tables.

Theorem 21: Boolean pc-tables are complete (hence pc-tables are also complete).

The previous theorem was independently observed in [20].

Theorem 22: pc-tables (and boolean pc-tables) are closed under the relational algebra.

The proof of this theorem gives in fact an algorithm for constructing the answer as a p-database itself, represented by a pc-table. In particular this will work for the models of [10, 16, 23] or for models we might invent by adding probabilistic information to v-tables or to the representation systems considered in [19]. The interesting result of [6] about the applicability of an "extensional" algorithm to calculating answer tuple probabilities can be seen also as characterizing the conjunctive queries q which for any p-?-table T are such that the c-table $\bar{q}(T)$ is in fact equivalent to some p-?-table.

8 Conclusion

We reviewed some old and some new examples of representation systems for incomplete and probabilistic databases. We discussed notions of expressive completeness, and we gave a new notion of completeness, called $\mathcal{RA}$-completeness, which makes sense in the case of infinite domains. We introduced the concept of algebraic completion and gave some results showing that extending weaker models by various fragments of the relational algebra yields complete models. Finally, we showed how probabilistic representation systems can be seen as probabilistic counterparts of incomplete representation systems, and as an example we proposed a probabilistic representation system called pc-tables, which we showed to be closed and complete.

References

[1] S. Abiteboul and O. M. Duschka. Complexity of Answering Queries Using Materialized Views. In *PODS*, pages 254–263, 1998.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison–Wesley, Reading, MA, 1995.

[3] M. Arenas, L. E. Bertossi, and J. Chomicki. Answer sets for consistent query answering in inconsistent databases. *TPLP*, 3(4-5):393–424, 2003.

[4] D. Barbara, H. Garcia-Molina, and D. Porter. A probabilistic relational data model. In *EDBT*, 1990.

[5] R. Cavallo and M. Pittarelli. The Theory of Probabilistic Databases. In *VLDB*, pages 71–81, 1987.

[6] N. Dalvi and D. Suciu. Efficient Query Evaluation on Probabilistic Databases. In *VLDB*, 2004.

[7] D. Dey and S. Sarkar. A Probabilistic Relational Model and Algebra. *ACM TODS*, 21(3):339–369, 1996.

[8] R. Durrett. *Probability: Theory and Examples*. Duxbury Press, 3rd edition, 2004.

[9] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT*, pages 207–224, London, UK, 2003. Springer-Verlag.

[10] N. Fuhr and T. Rölleke. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. *ACM TODS*, 14(1):32–66, 1997.

[11] E. Grädel, Y. Gurevich, and C. Hirch. The Complexity of Query Reliability. In *PODS*, 1998.

[12] G. Grahne. *The Problem of Incomplete Information in Relational Databases*, volume 554 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1991.

[13] T. J. Green and V. Tannen. Models for Incomplete and Probabilistic Information. In *IIDB*, 2006.

[14] T. Imieliński and W. Lipski, Jr. Incomplete Information in Relational Databases. *J. ACM*, 31(4), 1984.

[15] T. Imieliński, S. A. Naqvi, and K. V. Vadaparty. Incomplete objects — a data model for design and planning applications. In *SIGMOD*, pages 288–297, 1991.

[16] L. V. S. Lakshmanan, N. Leone, R. Ross, and V. S. Subrahmanian. ProbView: a Flexible Probabilistic Database System. *ACM TODS*, 22(3):419–469, 1997.

[17] L. Libkin and L. Wong. Semantic representations and query languages for or-sets. *J. Computer and System Sci.*, 52(1):125–142, 1996.

[18] F. Sadri. Modeling Uncertainty in Databases. In *ICDE*, pages 122–131. IEEE Computer Society, 1991.

[19] A. D. Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working Models for Uncertain Data. In *ICDE*, 2006.

[20] D. Suciu and N. Dalvi. Foundations of probabilistic answers to queries (tutorial). In *SIGMOD*, 2005.

[21] R. van der Meyden. Logical Approaches to Incomplete Information: A Survey. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*. Kluwer, 1998.

[22] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In *CIDR*, 2005.

[23] E. Zimányi. Query evaluation in probabilistic databases. *Theoretical Computer Science*, 171(1–2), 1997.