The Consistency of Probabilistic Databases with Independent Cells

Amir Gilad
Duke University, Durham, NC, USA

Aviram Imber
Technion – Israel Institute of Technology, Haifa, Israel

Benny Kimelfeld
Technion – Israel Institute of Technology, Haifa, Israel

Abstract

A probabilistic database with attribute-level uncertainty consists of relations where cells of some attributes may hold probability distributions rather than deterministic content. Such databases arise, implicitly or explicitly, in the context of noisy operations such as missing data imputation, where we automatically fill in missing values, column prediction, where we predict unknown attributes, and database cleaning (and repairing), where we replace the original values due to detected errors or violation of integrity constraints. We study the computational complexity of problems that regard the selection of cell values in the presence of integrity constraints. More precisely, we focus on functional dependencies and study three problems: (1) deciding whether the constraints can be satisfied by any choice of values, (2) finding a most probable such choice, and (3) calculating the probability of satisfying the constraints. The data complexity of these problems is determined by the combination of the set of functional dependencies and the collection of uncertain attributes. We give full classifications into tractable and intractable complexities for several classes of constraints, including a single dependency, matching constraints, and unary functional dependencies.

2012 ACM Subject Classification Information systems → Data management systems

Keywords and phrases Probabilistic databases, attribute-level uncertainty, functional dependencies, most probable database

Digital Object Identifier 10.4230/LIPIcs.ICDT.2023.22

Related Version *Full Version*: `https://arxiv.org/abs/2212.12104`

Funding The work of Amir Gilad was supported by NSF awards IIS-1703431, IIS-1552538, and IIS-2008107. The work of Aviram Imber and Benny Kimelfeld was supported by the Israel Science Foundation (ISF), Grant 768/19, and the German Research Foundation (DFG) Project 412400621 (DIP program).

1 Introduction

Various database tasks amount to reasoning about relations where attribute values are uncertain. To name a few, systems for *data cleaning* may detect errors and suggest alternative fixes with different confidence scores [15, 27, 28], approaches to *data repair* may suggest alternative values due to the violation of integrity constraints (e.g., key constraints and more general functional dependencies) [2, 34], and algorithms for *missing-data imputation* may suggest a probability distribution over possible completions of missing values [3, 24]. Such uncertainty is captured as a probabilistic database in the so called *attribute-level uncertainty* [29] (as opposed to the commonly studied *tuple-level uncertainty* [8]).

We refer to a relation of a probabilistic database in the attribute-level uncertainty as a Cell-Independent Relation (CIR). A CIR is a probabilistic database with a single relation, where the content of a cell is a distribution over possible values, and different cells are probabilistically

tid	room	?specialist	time
1	41	`Bart`(0.5) | `Lisa`(0.5)	`5 PM`
2	163	`Bart`(0.7) | `Lisa`(0.3)	`5 PM`
3	41	`Bart`(0.2) | `Maggie`(0.8)	`5 PM`

(a) CIR U_1 with uncertain specialist.

$F_1 :=$ {?specialist time → room}
$F_2 :=$ {?specialist time → room ,
room time → ?specialist}

(b) Sets F_1 and F_2 of functional dependencies.

tid	room	?specialist	time
1	41	`Lisa`	`5 PM`
2	163	`Bart`	`5 PM`
3	41	`Maggie`	`5 PM`

tid	room	?specialist	time
1	41	`Bart`	`5 PM`
2	163	`Lisa`	`5 PM`
3	41	`Bart`	`5 PM`

(c) Samples r (left) and r' (right) of U_1.

Figure 1 Running example: CIR, FDs, and samples.

independent. The CIR is the correspondent of a relation in the *Tuple-Independent Database* (TID) under the *tuple-level uncertainty*, where the existence of each tuple is uncertain (while its content is certain), and different tuples are probabilistically independent [29]. In contrast, the tuples of a CIR always exist, but their content is uncertain. For illustration, Figure 1a depicts a CIR with uncertain information about specialists attending rooms (e.g., since their attendance is determined by noisy sensors). Some attributes (here room and business) are certain and have deterministic values. The uncertain attributes (e.g., ?specialist) are marked by a question mark and their cells have several options for values. We later explain how this distinction has a crucial impact on the complexity of CIRs.

A natural scenario, studied by previous work for the TID model [12, 21], considers a probabilistic database in the presence of a given set of integrity constraints, and specifically, Functional Dependencies (FDs). Such a scenario gives rise to several interesting computational challenges, and we focus here on three basic ones. In the problem of *possible consistency*, the goal is to test for the existence of a possible world (with a nonzero probability) that satisfies the FDs. The problem of the *most probable database* ("MPD" [12]) is that of finding a possible world that satisfies the FDs and has the highest probability. In the problem of computing the *probability of consistency*, the goal is to calculate the above probability exactly (beyond just deciding whether it is nonzero), that is, the probability that (a random sample of) the given CIR satisfies the underlying FDs. We investigate the computational complexity of these three problems for the CIR model. Our results provide classifications of tractability for different classes of FDs. Importantly, we show that, for the studied classes, the complexity of these problems is determined by two factors: *(1)* the location of the uncertain attributes in the FDs (left or right side), *and (2)* the combination of the FDs in the given set of constraints.

The three problems relate to each other in the following manner. To solve MPD, we need to be able to solve the possible consistency problem. The analysis of the probability of consistency sheds light on the possible consistency problem (is it fundamentally harder to compute the probability than to just determine whether it is nonzero?), but its importance goes beyond that. As we explain in Section 4, computing this probability is useful to any type of constraint over CIRs, as the tractability of this probability implies that we can efficiently sample correctly from the conditional space of consistent samples.

Our study adopts the standard yardstick of *data complexity* [33], where we fix the relational schema and the set of functional dependencies. The schema mentions not only what attributes are in the header of the relation, but also *which attribute is certain and* *which attribute is uncertain.* The complexity of the problems can be different for different combinations of schema and constraints, and we aim for a detailed understanding of which combinations are tractable and which are not.

▶ **Example 1.** Consider again the CIR U_1 in Figure 1a along with the FD set F_1 of Figure 1b, consisting of a single FD. The FD says that, at a specific time, a specialist can be found in only one location. Figure 1c shows a consistent sample r of U_1 whose probability is $\Pr(r) = 0.5 \cdot 0.7 \cdot 0.8 = 0.28$. In particular, this probability is nonzero and, so, U_1 is possibly consistent. This sample has a maximal probability among the consistent samples; therefore, r is a most probable database for U_1. Now, consider the FD set F_2 shown in Figure 1b, where the first FD is the one of F_1 and the second states that no two specialists should be in the same room at the same time. The sample r in Figure 1c is no longer consistent, but r' (in the same figure) is a consistent sample and also a most probable database. In fact, r' is the only consistent sample in this case, so the probability of consistency for F_2 turns out to be that of r'. ⌟

In contrast to the state of affairs for the attribute-level uncertainty, much is known about the complexity of MPD in the case of *tuple-level uncertainty* (i.e., finding the most likely instance of a tuple-independent probabilistic database conditioned on conformance to a set of FDs). As we explain later in the discussion on related work, past research has established a full classification of the complexity of the sets of FDs into tractable and intractable instances of MPD. In this work, we aim to bring our understanding of attribute-level uncertainty closer to tuple-level uncertainty.

Results. We would like to understand the complexity of every scenario defined by a schema and set of FDs, and in particular, establish a dichotomy that charts the exact conditions that cast each problem tractable. This, however, remains open for future investigation. Yet, we make considerable progress towards that. We establish classification results on several classes of functional dependencies:

- Singleton FDs (i.e., FD sets with a single FD);
- Matching constraints (i.e., FD sets of the form $\{X \to Y, Y \to X\}$ for arbitrary X and Y);
- Arbitrary sets of *unary* FDs (i.e., FDs with a single attribute on the left side).

Each classification consists of three internal classifications – one for each of the three problems we study (possible consistency, most probable database, and the probability of consistency). In every case, finding a most probable database turns out to be tractable whenever possible consistency is tractable. There are cases where the probability of consistency is intractable in contrast to the tractability of the most probable database, but we did not find any case where the other direction holds (and we will be surprised if such a case exists). We also establish some general conclusions beyond these classes. For example, in Theorem 14 (of Section 5) we claim that if we make no assumption that some attributes are certain, then possible consistency is hard for *every nontrivial set of FDs.*

▶ **Example 2.** Reconsider the CIR U_1 in Figure 1a along with the FD set F_1 of Figure 1b, consisting of a single FD. Our classification shows that, in general, finding a solution to the possible consistency problem for such an FD, with uncertain attributes on the left side, is NP-complete. Now, reconsider the FD set F_2 shown in Figure 1b, where the first FD is the one of F_1. Thus, $F_1 \subset F_2$, however, interestingly, our results show that for sets with the structure of F_2, finding an MPD (and, hence, also solving possible consistency) is in polynomial time. Intuitively, the additional FD in F_2 constrains the uncertain attribute on the left side of the first FD, making the problem tractable. Finally, computing the probability of consistency for sets with the structure of F_1 and F_2 is #P-hard (or more precisely $\mathrm{FP}^{\#\mathrm{P}}$-complete). ⌟

Related work. In the case of tuple-independent databases, Gribkoff, Van den Broeck, and Suciu [12] established a dichotomy in the complexity of MPD for sets of unary FDs. This dichotomy has been generalized by Livshits, Kimelfeld and Roy [21] to a full classification over all sets of FDs, where they also established that the problem is equivalent to finding a *cardinality repair* of an inconsistent database. Carmeli et al. [5] showed that two tractable cases, namely a *single FD* and a *matching constraint*, remain tractable even if the FDs are treated as *soft constraints* (where every violation incurs a cost).

A most probable database is the same as the "Most Likely Intention" (MLI) in the framework of Probabilistic Unclean Databases (PUD) of De Sa et al. [28], in the special case where the *intention model* demands hard constraints and the *realization model* applies random changes to cells independently in what they refer to as *parfactor/update* PUD. They showed that finding an MLI of a parfactor/update PUD generalizes the problem of finding an *update repair* of an inconsistent database with a minimum number of value changes. In turn, finding a minimal update repair has been studied in the literature and several complexity results are known for special cases of FDs, such as hardness (e.g., for the FD set $\{A \rightarrow B, B \rightarrow C\}$ due to Kolahi and Lakshmanan [18]) and tractability (e.g., for lhs-chains such as $\{A \rightarrow B, AD \rightarrow C\}$ due to Livshits et al. [21]). There are, though, substantial differences between finding a most probable consistent sample of a CIR and finding an optimal update repair of an inconsistent database, at least in the variations where complexity results are known. First, they allow to select *any* value (from an infinite domain) for a cell, in contrast to the distributions of the CIR that can limit the space of allowed values; indeed, this plays a major role in past repair algorithms (e.g., Proposition 5.6 of [21] and Algorithm FindVRepair of [18]). Second, they allow to change the value of *any* attribute and do not distinguish between uncertain attributes (where changes are allowed) and certain ones, as we do here; this is critical since, again, without such assumptions the problem is intractable for every nontrivial set of FDs (Theorem 14).

The problem of possible consistency does not have a nontrivial correspondence in the tuple-independent database model since, there, if there is any consistent sample then the subset that consists of all deterministic tuples (i.e., ones with probability one) is such a sample. The probability of consistency might be reminiscent of the problem of *repair counting* that was studied for subset repairs [4, 22]. Besides the fact that subset repairs are about tuple-level uncertainty (and no probabilities are involved), here we do not have any notion of *maximality* (while a repair is required to be a maximal consistent subset).

A CIR can be easily translated into a relation of a *block-independent-disjoint* (BID) probabilistic database [26]. In a BID, every relation is partitioned into independent blocks of mutually exclusive tuples, each associated with a probability. This model has also been studied under the terms *dirty database* [2] and x-tuples [6, 23, 25]. This translation implies that every upper bound for BIDs applies to CIRs, and the contrapositive: every hardness result that we establish (e.g., for the most probable database) extends immediately to BIDs; yet, it does not imply the other direction. Moreover, we are not aware of any positive results on inference over BIDs regarding integrity constraints. In addition, the translation from a CIR to a BID loses the information of which attributes are certain and which are uncertain, and as aforesaid, if we allow every attribute to be uncertain then the problem is hard for every nontrivial set of FDs (Theorem 14).

Organization. The remainder of the paper is organized as follows. We begin with preliminary definitions and notation (Section 2). We then define the CIR data model (Section 3) and the computational problems that we study (Section 4). Next, we describe our analysis for the case of singleton and matching dependencies (Section 5), and then the case of unary functional dependencies (Section 6). Lastly, we give concluding remarks (Section 7). Missing proofs can be found in the full version of the paper [10].

2 Preliminaries

We begin with preliminary definitions and notation.

Relations. We assume countably infinite sets **Val** of values and **Att** of attributes. A *relation schema* is a finite set $R = \{A_1, \dots, A_k\}$ of attributes. An *R-tuple* is a function $t : R \to \mathbf{Val}$ that maps each attribute $A \in R$ to a value that we denote by $t[A]$. A *relation* r is associated with a relation schema, denoted $\mathbf{Att}(r)$, a finite set of tuple identifiers, denoted $\mathsf{tids}(r)$, and a mapping from $\mathsf{tids}(r)$ to $\mathbf{Att}(r)$-tuples. (Note we allow for duplicate tuples, as we do not assume that the tuples of different identifiers are necessarily different.) We say that r is a relation *over* the relation schema $\mathbf{Att}(r)$. We denote by $r[i]$ the tuple that r maps to the identifier i. Hence, $r[i][A]$ is the value that tuple i has for the attribute A. As an example, Figure 1c (left) depicts a relation r with $\mathbf{Att}(r) = \{\mathsf{room}, ?\mathsf{specialist}, \mathsf{time}\}$ (for now, the question mark in ?specialist should be ignored.) Here, $\mathsf{tids}(r) = \{1, 2, 3\}$ and $r[1][\mathsf{room}] = \mathtt{41}$.

Suppose that X is a set of attributes. We denote by $\pi_X r$ the projection of r onto X. More precisely, $\pi_X r$ is the relation r' such that $\mathbf{Att}(r') = X$, $\mathsf{tids}(r') = \mathsf{tids}(r)$, and $r'[i][A] = r[i][A]$ for every $A \in \mathbf{Att}(r) \cap X$. Observe that in our notation, $(\pi_X r)[i]$ is the projection of tuple i to X. As a shorthand notation, we write $r[i][X]$ instead of $(\pi_X r)[i]$. For example, in Figure 1c we have $r[2][\mathsf{room}\ ?\mathsf{specialist}] = (\mathtt{163}, \mathtt{Bart})$.

Functional dependencies. A *functional dependency*, or FD for short, is an expression of the form $X \to Y$ where X and Y are finite sets of attributes. We say that $X \to Y$ is *over* a relation schema R if R contains all mentioned attributes, that is, $X \cup Y \subseteq R$. A relation r satisfies the FD $X \to Y$ over $\mathbf{Att}(r)$ if every two tuples that agree on X also agree on Y. In our notation, we say that r satisfies $X \to Y$ if for every two tuple identifiers i and i' in $\mathsf{tids}(r)$ it holds that $r[i][Y] = r[i'][Y]$ whenever $r[i][X] = r[i'][X]$. A relation r satisfies a set F of FDs over $\mathbf{Att}(r)$, denoted $r \models F$, if r satisfies every FD in F.

We use the standard convention that in instances of X and Y we may remove curly braces and commas. To compactly denote a set of FDs, we may also intuitively combine multiple FD expressions and change the direction of the arrows. For example, the notation $A \leftrightarrow B \leftarrow CD$ is a shorthand notation of $\{A \to B, B \to A, CD \to B\}$.

An FD $X \to Y$ is *unary* if X consists of a single attribute, and it is *trivial* if $Y \subseteq X$ (i.e., it is satisfied by every relation). A *matching constraint* (as termed in past work [5]) is a constraint of the form $X \leftrightarrow Y$, that is, the set $\{X \to Y, Y \to X\}$.

The *closure* F^+ of a set F of FDs is the set of all FDs that are implied by F (or, equivalently, can be inferred by repeatedly applying the axioms of Armstrong). For example, F^+ includes all of the trivial FDs. The closure X_F^+ of a finite set X of attributes is the set of all attributes A such that $X \to A$ is in F^+. Two finite attribute sets X and Y are *equivalent* (w.r.t. F) if $X_F^+ = Y_F^+$, or in other words, $X \to Y$ and $Y \to X$ are both in F^+. By a slight abuse of notation, we say that two attributes A and B are *equivalent* if $\{A\}$ and $\{B\}$ are

tid	business	?spokesperson	?location
1	`S. Propane`	`Mangione`(0.6) | `Strickland`(0.4)	`Arlen`(0.6) | `McMaynerberry`(0.4)
2	`Mega Lo Mart`	`Mangione`(0.45) | `Thatherton`(0.55)	`Arlen`(0.5) | `McMaynerberry`(0.5)
3	`Mega Lo Mart`	`Mangione`(0.4) | `Buckley`(0.6)	`Arlen`(0.55) | `McMaynerberry`(0.45)
4	`Get In Get Out`	`Peggy`(1.0)	`Arlen`(0.35) | `McMaynerberry`(0.3) | `Dallas`(0.35)

Figure 2 CIR U_2 with spokesperson and location as the uncertain attributes.

equivalent. Finally, if F is a set of FDs, then we denote by $\mathbf{Att}(F)$ the set of all attributes that occur in either the left or right sides of rules in F.

Probability distributions. We restrict our study in this paper to finite probability spaces (Ω, π) where Ω is a nonempty finite set of *samples* and $\pi : \Omega \to [0, 1]$ is a probability function satisfying $\sum_{o \in \Omega} \pi(o) = 1$. The *support* of $\delta = (\Omega, \pi)$, denoted $\mathsf{supp}(\delta)$, is the set of samples $o \in \Omega$ such that $\pi(o) > 0$. We denote by $\Pr_\delta(o)$ the probability $\pi(o)$. We may write just $\Pr(o)$ when δ is clear from the context.

3 Cell-Independent Relations

A *Cell-Independent Relation*, or *CIR* for short, is similar to an ordinary relation, except that in certain attributes the values may be probabilistic; that is, instead of an ordinary value, each of them contains a probability distribution over values. One could claim that the model should allow every attribute to have uncertain values. However, knowing which attributes are certain has a major impact on the complexity of operations over CIRs. Formally, a CIR U is defined similarly to a relation, with the following differences:

- The schema of U, namely $\mathbf{Att}(U)$, has *marked attributes* where uncertain values are allowed. We denote a marked attribute using a leading question mark, as in $?A$, and the set of marked attributes by $?\mathbf{Att}(U)$. (Note that $?\mathbf{Att}(U)$ is a subset of $\mathbf{Att}(U)$.)
- For every $i \in \mathsf{tids}(U)$ and marked attribute $?A \in ?\mathbf{Att}(U)$, the cell $U[i][?A]$ is a probability distribution over $\mathbf{Val}$.

By interpreting cells as probabilistically independent, a CIR U represents a probability distribution over ordinary relations. Specifically, a sample of U is a relation that is obtained from U by sampling a value for each uncertain cell. More formally, a sample of U is a relation r such that $\mathbf{Att}(r) = \mathbf{Att}(U)$, $\mathsf{tids}(r) = \mathsf{tids}(U)$, and for every $i \in \mathsf{tids}(r)$ and unmarked attribute A we have that $r[i][A] = U[i][A]$.

The probability $\Pr_U(r)$ of a sample r of U is the product of the probabilities of the values chosen for r:

$$\Pr_U(r) = \prod_{i \in \mathsf{tids}(U)} \prod_{?A \in ?\mathbf{Att}(U)} \Pr_{U[i][?A]}(r[i][?A])$$

Note that $\Pr_{U[i][?A]}(r[i][?A])$ is the probability of the value $r[i][?A]$ (i.e., the value that tuple i of r has for the attribute $?A$) according to the distribution $U[i][?A]$ (i.e., the distribution that tuple i of U has for the attribute $?A$).

▶ **Example 3.** Figures 1a and 2 depict examples U_1 and U_2, respectively, of CIRs. U_1 has been discussed in Example 1 and U_2 describes a CIR that stores businesses along with their spokespeople and headquarters locations. Some information in U_2 is noisy (e.g., since the rows are scraped from Web pages), and particularly the identity of the spokesperson and the business location. U_1 has a single uncertain attribute, namely ?$\mathsf{specialist}$, and U_2 has two uncertain attributes, namely ?$\mathsf{spokesperson}$ and ?$\mathsf{location}$. In particular, we have:

$$\mathbf{Att}(U_1) = \{\mathsf{room}, ?\mathsf{specialist}, \mathsf{time}\} \qquad ?\mathbf{Att}(U_1) = \{?\mathsf{specialist}\}$$

Distributions over values are written straightforwardly in the examples. For example, the distribution $U_1[2][?\mathsf{specialist}]$ is the uniform distribution that consists of `Bart` and `Lisa`, each with probability 0.5.

The relations r and r' of Figure 1c are samples of U_1. By the choices made in r, the probability $\Pr_U(r)$ is $0.5\cdot 0.7\cdot 0.8$. Note that the probability of r is smaller than the probability of the sample where the specialists are `Lisa`, `Bart` and `Maggie`, for instance, respectively. ⌟

Simplified notation. In the analyses that we conduct in later sections, we may simplify the notation when defining a CIR U. When $\mathbf{Att}(U) = \{A_1, \dots, A_k\}$, we may introduce a new tuple $t[i]$ with $t[i][A_\ell] = a_\ell$ simply as $(a_1, \dots, a_k)$, assuming that the attributes are naturally ordered alphabetically by their symbols. For example, if $\mathbf{Att}(U) = \{A, B, C\}$, then (a, b, c) corresponds to the tuple that maps A, B and C to a, b and c, respectively. We can also use a distribution δ instead of a value a_ℓ. In particular, we write $b_1|\dots|b_t$ to denote a uniform distribution among the values $b_1, \dots, b_t$.

▶ **Example 4.** Continuing Example 3, in the simplified notation the tuple $U_1[2]$ can be written as $(\mathtt{163}, \mathtt{Bart}|\mathtt{Lisa}, \mathtt{5\ pm})$ since the attributes are ordered lexicographically and, again, the distribution happens to be uniform. ⌟

Consistency of CIRs

Let F be a set of FDs and let U be a CIR, both over the same schema. A *consistent sample* of U is a relation $r \in \mathsf{supp}(U)$ such that $r \models F$. We say that U is *possibly consistent* if at least one consistent sample exists. By *the probability of consistency*, we refer to the probability $\Pr_{r\sim U}(r \models F)$ that a random sample of U satisfies F. As a shorthand notation, we denote this probability by $\Pr_U(F)$. Note that U is possibly consistent if and only if $\Pr_U(F) > 0$. A consistent sample r is a *most probable database* (using the terminology of Gribkoff, Van den Broeck and Suciu [12]) if $\Pr(r) \geq \Pr(r')$ for every other consistent sample r'.

▶ **Example 5.** Consider the CIR U_1 of Figure 1a. Let F_1 be that of Figure 1b, saying that at a specific time, a specialist can be found in only one location. Figure 1c (left) shows a consistent sample r of U_1. Then $\Pr(r) = 0.5 \cdot 0.7 \cdot 0.8 = 0.28$. In particular, this probability is nonzero, hence U_1 is possibly consistent. The reader can verify that r has a maximal probability among the consistent samples (and, in fact, among all samples); therefore, r is a most probable database for U_1. To calculate the probability of consistency, we will take the complement of the probability of *inconsistency*. An inconsistent sample can be obtained in two ways: *(1)* selecting `Lisa` in both the first and second tuples, *or (2)* selecting `Bart` in the second tuple and in at most one of the first and the third (which we can compute as the complement of the product of the probabilities of selecting the others). Therefore,

$$\Pr_{U_1}(F_1) = 1 - \big(0.3 \cdot 0.5 + 0.7 \cdot (1 - 0.5 \cdot 0.8)\big)\,.$$

Now suppose that we use F_2 of Figure 1b saying that, in addition to F_1, a room can host only one specialist at a specific time. In this case, r is no longer a consistent sample since Room `41` hosts different specialists at `5 PM`, namely `Lisa` and `Maggie`. The reader can verify that the only consistent sample now is r' of Figure 1c. In particular, U_1 remains possible consistent, the sample r' is the most probable database, and the probability of consistency is the probability of r', namely $0.5 \cdot 0.3 \cdot 0.2$. ⌟

4 Consistency Problems

We study three computational problems in the paper, as in the following definition.

▶ **Definition 6.** *Fix a schema R and a set F of FDs over R. In each of the following problems, we are given as input a CIR U over R:*

1. **Possible consistency:** *determine whether $\Pr_U(F) > 0$.*
2. **Most probable database:** *find a consistent sample with a maximum probability.*
3. **Probability of consistency:** *calculate $\Pr_U(F)$.*

Observe that these problems include the basics of probabilistic inference: *maximum likelihood* computation and *marginal probability* calculation. An MPD can be viewed as an optimal completion of missing values, or an optimal correction of values suspected of being erroneous, assuming the independence of cells (as a *prior* distribution) and conditioned on satisfying the constraints (as a *posterior* distribution). A necessary condition for the tractability of the most probable database is possible consistency, where we decide whether at least one consistent sample exists. The problem of computing the probability of consistency can be thought of as a basic problem that sheds light on possible consistency. For example, if possible consistency is decidable in polynomial time in some case, is it because we can, generally, compute the probability of consistency or because there is something fundamentally easier with feasibility? We will see cases that feature both phenomena.

A more technical reason for computing the probability of consistency is that it provides the ability to *sample* soundly from the conditional probability distribution (the posterior). More precisely, an efficient algorithm for computing the probability of consistency can be used to devise an efficient randomized algorithm that produces a consistent sample r with the probability $\Pr_U(r \mid r \models F)$. The idea is quite simple and applies to every condition F over databases, regardless of being FDs (and was used in different settings, e.g., [7]).

As aforesaid, the second and third problems are at least as hard as the first one: finding a most probable database of U requires knowing whether U is possibly consistent, and calculating the exact probability is at least as hard as determining whether it is nonzero. There is no reason to believe a-priori that the complexities of the second and third problems are comparable. Yet, our analysis will show that the third has the same or higher complexity in the situations that we study.

4.1 Complexity Assumptions

In our complexity analysis, we will restrict the discussion to uncertain cells that are finite distributions represented explicitly by giving a probability for each value in the support. Note that if all uncertain cells of U have a finite distribution, then U has a finite set of samples. Yet, its size can be exponential in the number of rows of U (and also in the number of columns of U, though we will treat this number as fixed as we explain next), even if each cell distribution is binary (i.e., has only two nonzero options). Every probability is assumed to be a rational number that is represented using the numerator and the denominator.

We will focus on the *data complexity* of problems, which means that we will make the assumption that the schema R of the CIR and the set F of FDs are both fixed. Hence, every combination (R, F) defines a separate computational problem, and different pairs (R, F) can potentially have different complexities.

Table 1 Complexity of the consistency problems for a binary schema. "Possibility" refers to possible consistency, "MPD" refers to the most probable database problem, and "Probability" refers to the probability of consistency.

FDs	Possibility	MPD	Probability	Results
$A \to ?B$	PTime	PTime	PTime	Proposition 7
$?A \to B$	NP-complete	NP-hard	$\mathrm{FP}^{\#\mathrm{P}}$-complete	Lemma 10
$A \leftrightarrow ?B$	PTime	PTime	$\mathrm{FP}^{\#\mathrm{P}}$-complete	Prop. 8 (PTime), 9 ($\mathrm{FP}^{\#\mathrm{P}}$-c.)
$?A \leftrightarrow ?B$	NP-complete	NP-hard	$\mathrm{FP}^{\#\mathrm{P}}$-complete	Lemma 11

4.2 Preliminary Observations

In the following sections, we study the complexity of the three consistency problems that we defined in Definition 6. Before we move on to the actual results, let us state some obvious general observations.

- Possible consistency is in NP, since we can verify a "yes" instance U in polynomial time by verifying that a relation r is a consistent sample.
- If possible consistency is NP-complete for some schema R and set F of FDs, then it is NP-hard to find a most probable database, and it is NP-hard to compute the probability of consistency.
- We will show that the probability of consistency can be #P-hard, or more precisely $\mathrm{FP}^{\#\mathrm{P}}$-complete.[1] Membership in $\mathrm{FP}^{\#\mathrm{P}}$ of the probability of consistency is based on our assumption that probabilities are represented as rational numbers, and it can be shown using standard techniques (e.g., [1, 11]) that we do not repeat here.

We will take the above for granted and avoid repeating the statements throughout the paper.

5 Singleton and Matching Constraints

In this section, we investigate the complexity of the three problems we study in two special cases: a singleton constraint $\{X \to Y\}$ and a *matching constraint* $X \leftrightarrow Y$ (as it has been termed in past work [5]). We give full classifications of when such constraints are tractable and intractable for the three problems. We note that we leave open the classification of the entire class of FD sets, but we provide it for the general case of unary FDs in Section 6.

We begin with the case of a binary schema, where every set of FDs is equivalent to either a singleton or a matching constraint.

5.1 The Case of a Binary Schema

Throughout this section, we assume that the schema is $\{A, B\}$. The complexity of the different cases of FDs is shown in Table 1. To explain the entries of the table, let us begin with the tractable cases.

[1] Recall that $\mathrm{FP}^{\#\mathrm{P}}$ is the class of functions that are computable in polynomial time with an oracle to a problem in #P (e.g., counting the number of satisfying assignments of a propositional formula). This class is considered intractable, and above the polynomial hierarchy [30].

5.1.1 Algorithms

In this section, we show algorithms for $A \rightarrow ?B$ and for $A \leftrightarrow ?B$.

For $A \rightarrow ?B$, we need to determine a value b for each value a of the attribute A. The idea is that we do so independently for each a. Let V_A be the active domain of the attribute A of U, and V_B be the set of all values in the supports of the distributions of B. Formally:

$$V_A := \{U[i][A] \mid i \in \mathsf{tids}(U)\} \qquad V_B := \bigcup \{\mathsf{supp}(U[i][B]) \mid i \in \mathsf{tids}(U)\}$$

A consistent sample r selects a value $b_a \in V_B$ for each $a \in V_A$, and then $\Pr_U(r) = \prod_{a \in V_A} p(a, b_a)$ where $p(a,b)$ is given by:

$$p(a,b) := \prod_{i:U[i][A]=a} \Pr_{U[i][B]}(b)$$

Therefore, to find a most probable database, we consider each $a \in V_A$ independently, and find a $b \in V_B$ that maximizes $p(a,b)$. This b will be used for the tuples with the value a in A. In addition, we have the following formula that gives us immediately a polynomial-time algorithm (via a direct computation) for the probability of consistency:

$$\Pr_U(\{A \rightarrow B\}) = \prod_{a \in V_A} \sum_{b \in V_B} p(a,b)$$

Where $\sum_{b \in V_B} p(a,b)$ is the probability that the tuples with the value a for A agree on their B attribute. In summary, we have established the following.

▶ **Proposition 7.** *All three problems in Definition 6 are solvable in polynomial time for $A \rightarrow ?B$.*

Next, we discuss $A \leftrightarrow ?B$. Let U be a CIR. A consistent sample r of U entails the matching of each A value a to each B value b, so that no two a values occur with the same b, and no two bs occur with the same a. Therefore, we can solve this problem using an algorithm for minimum-cost perfect matching, as follows. Let V_A, V_B and $p(a,b)$ be as defined earlier in this section for $A \rightarrow ?B$. We construct a complete bipartite graph G as follows.

- The left-side vertex set is V_A and the right-side vertex set is V_B.
- The cost of every edge (a,b) is $(-\log p(a,b))$; we use this weight as as our goal is to translate a maximum product into a minimum sum.[2]

Note that $|V_A|$ and $|V_B|$ are not necessarily of the same cardinally. If $|V_A| > |V_B|$, then U has no consistent sample at all. If $|V_A| < |V_B|$, then we add to the left side of the graph dummy vertices a' that are connected to all V_B vertices using the same cost, say 1. With this adjustment, we can now find a most probable database by finding a minimum-cost perfect matching in G (e.g., with the Hungarian method [19]). In summary, we have established the following.

▶ **Proposition 8.** *For $A \leftrightarrow ?B$, a most probable database can be found in polynomial time.*

It turns out that the third problem, the probability of consistency, is intractable. We show it in the next section.

[2] We assume that the computational model for finding a minimum-cost perfect matching can handle the representation of logarithms, including $\log 0 = -\infty$. As an alternative, we could use directly an algorithm for maximizing the product of the edges in the perfect matching [31].

5.1.2 Hardness

We now discuss the hardness results of Table 1. We begin with $A \leftrightarrow ?B$. Recall that possible consistency and the most probable database are solvable in polynomial time (Proposition 8). The probability of consistency, however, is hard.

▶ **Proposition 9.** *For $A \leftrightarrow ?B$, it is $\mathrm{FP}^{\#\mathrm{P}}$-complete to compute the probability of consistency.*

Proof. We show a reduction from the problem of counting the perfect matchings of a bipartite graph (which is the same as calculating the permanent of a 0/1-matrix). This problem is known to be #P-complete [32]. We are given a bipartite graph $G = (V_L, V_R, E)$ such that $|V_L| = |V_R|$ and the goal is to compute the number of perfect matchings that G has. We construct a CIR U as follows. For each vertex $v \in V_L$ we collect the set $N_v \subseteq V_R$ of neighbors of v. Let $N_v = \{u_1, \dots, u_\ell\}$. We add to U the tuple $(v, u_1|\dots|u_\ell)$.

Observe that every consistent sample induces a perfect matching (due to $A \leftrightarrow ?B$), and vice versa. Hence, the number of consistent samples of U is the same as the number of perfect matchings of G. Since we used only uniform probabilities, every sample of U has the same probability, namely $1/(\prod_{v\in V_L} |N_v|)$. Therefore, the number of perfect matchings is $\Pr_U(A \leftrightarrow ?B) \cdot \prod_{v\in V_L} |N_v|$. ◀

The next two lemmas address the case of $?A \to B$ and the case of $?A \leftrightarrow ?B$, respectively. We begin with $?A \to B$.

▶ **Lemma 10.** *For $?A \to B$:*
1. *Possible consistency is NP-complete.*
2. *It is $\mathrm{FP}^{\#\mathrm{P}}$-complete to compute the probability of consistency.*

Proof. We prove each part separately.

Part 1. We show a reduction from *non-mixed satisfiability* (NM-SAT), where each clause contains either only positive literals ("positive clause") or only negative literals ("negative clause"). This problem is known to be NP-complete [13].

We are given a formula $c_1 \wedge \cdots \wedge c_m$ over $x_1, \dots, x_n$. We construct an uncertain table as follows. For each positive $c_i = y_1 \vee \cdots \vee y_\ell$ we have in the table the tuple

$$(y_1|\dots|y_\ell, \mathbf{true}),$$

that is, a tuple with a distinct identifier i such that $U[i][A]$ is a uniform distribution over $\{y_1, \dots, y_\ell\}$ and $U[i][B]$ is the value **true**. Similarly, for each negative clause $c_i = \neg y_1 \vee \cdots \vee \neg y_\ell$ we have in the table the tuple

$$(y_1|\dots|y_\ell, \mathbf{false}).$$

Hence, for each positive clause we need to select one satisfying variable, for each negative clause we need to select one satisfying variable, and we cannot select the same variable to satisfy both a positive and a negative clause. This immediately implies the correctness of the reduction.

Part 2. To prove Part 2, we use a reduction from counting the perfect matchings, similarly to the proof of Proposition 9, except that now we reverse the order of the attributes: Instead of adding the tuple $(v, u_1|\dots|u_\ell)$, we add the tuple $(u_1|\dots|u_\ell, v)$. The reader can easily verify that each consistent sample again encodes a unique perfect matching, and vice versa. ◀

We now move on to $?A \leftrightarrow ?B$.

▶ **Lemma 11.** *For $?A \leftrightarrow ?B$:*

1. *Possible consistency is NP-hard.*
2. *It is* $\mathrm{FP}^{\#\mathrm{P}}$*-complete to compute the probability of consistency.*

Proof. We prove each part separately.

Part 1. We need to show the NP-hardness of possible consistency. We show a reduction from standard SAT, where we are given a formula $\varphi = c_1 \wedge \cdots \wedge c_m$ over $x_1, \dots, x_n$, and we construct a CIR U over $\{A, B\}$ as follows. For each clause $c = d_1 \vee \cdots \vee d_\ell$ we add to U the tuple

$$(c, \langle c, d_1\rangle | \dots | \langle c, d_\ell\rangle) \,.$$

Note that the values of U are clauses c and pairs $\langle c, d\rangle$ where d is a literal. In addition to these tuples, we collect every two pairs $\langle c, d\rangle$ and $\langle c', d'\rangle$ such that d and d' are in conflict, that is, if $d = x$ then $d' = \neg x$ and if $d = \neg x$ then $d' = x$. For each such pair, we add to U the tuple

$$(\langle c, d\rangle | \langle c', d'\rangle, \langle c, d\rangle | \langle c', d'\rangle) \,.$$

This completes the reduction. Next, we prove the correctness of the reduction, that is, φ is satisfiable if and only if U is possibly consistent.

For the "only if" direction, suppose that τ is a satisfying truth assignment for φ. We construct a consistent sample r as follows. For every tuples of the form $(c, \langle c, d_1\rangle | \dots | \langle c, d_\ell\rangle)$, we choose for B a value $\langle c, d_i\rangle$ such that $\tau(d_i) = \textbf{true}$. In the case of tuples of the form $(\langle c, d\rangle | \langle c', d'\rangle, \langle c, d\rangle | \langle c', d'\rangle)$, we choose the pair $\langle c', d'\rangle$ such that $\tau(d') = \textbf{false}$ for both attributes A and B. We need to show that r satisfies $?A \leftrightarrow ?B$. It is easy to see why the left attribute determines the right attribute, and so, $?A \rightarrow ?B$ holds. Regarding $?B \rightarrow ?A$, we need to verify that we do not have any conflicting tuples $(c, \langle c, d\rangle)$ and $(\langle c', d'\rangle, \langle c', d'\rangle)$ where $c = c'$ and $d = d'$. This is due to the fact that $\tau(d) = \textbf{true}$ and $\tau(d') = \textbf{false}$.

For the "if" direction, suppose that r is a consistent sample. We define a satisfying truth assignment τ as follows. Suppose that r contains $(c, \langle c, d\rangle)$. Then r necessarily contains $(\langle c', d'\rangle, \langle c', d'\rangle)$ for every c' that contains the negation d' of d. Therefore, r does not contain any $(c', \langle c', d'\rangle)$ where d' contradicts d. So, we choose τ such that $\tau(d) = \textbf{true}$. If needed, we complete τ to the remaining variables arbitrarily. From the construction of τ it holds that every clause c is satisfied. This completes the proof of Part 1.

Part 2. Note that this part follows immediately from Proposition 9, since every instance of $A \leftrightarrow ?B$ can be viewed as an instance of $?A \leftrightarrow ?B$ where all A values are known. ◀

We have now completed all results of Table 1. We will use these results for the extension to singleton, matching, and unary constraints.

5.2 Beyond Binary Schemas

We generalize the results for the binary case to the more general case where the FD set is either a singleton or a matching constraint and the schema can have more than two attributes.

▶ **Theorem 12.** *Let X and Y be sets of attributes such that $X \not\subseteq Y$ and $Y \not\subseteq X$, and at least one attribute in $X \cup Y$ is uncertain.*

1. *In the case of $X \to Y$: If X consists of only certain attributes, then all three problems are solvable in polynomial time; otherwise, possible consistency is NP-complete and the probability of consistency is $\mathrm{FP}^{\#\mathrm{P}}$-complete.*
2. *In the case of $X \leftrightarrow Y$: If either X or Y consists of only certain attributes, then a most probable database can be found in polynomial time; otherwise, possible consistency is NP-hard. In any case, the probability of consistency is $\mathrm{FP}^{\#\mathrm{P}}$-complete.*

Proof sketch. For the first part, the tractability side is via a reduction to the case of $A \to ?B$, which is tractable due to Proposition 7. The hardness side is due to a straightforward reduction from $?A \to B$, where hardness is stated in Lemma 10. For the second part, the tractability side is via a reduction to the case of $A \leftrightarrow ?B$, which is tractable due to Proposition 8. The hardness of possible consistency relies on the cases of $?A \to B$ and $?A \leftrightarrow ?B$ from Lemma 10 and Lemma 11, respectively. ◀

▶ **Example 13.** Consider again the CIR U_1 of Figure 1a, and the following two constraints: $F_1 := \{$?specialist time $\to$ room$\}$ and $F_2 := F_1 \cup \{$room time $\to$?specialist$\}$. For F_1, all three problems are hard, since the left hand side of the FD contains the uncertain attribute ?specialist. For F_2, a most probable database can be found in polynomial time, since F_2 is equivalent to room time $\leftrightarrow$?specialist time, where one side (the left side) consists of only certain attributes. However, the probability of consistency remains $\mathrm{FP}^{\#\mathrm{P}}$-hard. ⌟

Note that in Theorem 12, the assumption that $X \not\subseteq Y$ and $Y \not\subseteq X$ does not lose generality, for the following reason. If $X \subseteq Y$, then the FD $X \to Y$ is equivalent to $X \to Y \setminus X$, the FD $Y \to X$ is trivial, and the matching constraint $X \leftrightarrow Y$ is equivalent to the singleton $\{X \to Y\}$ (which is covered in Part 1).

From Theorem 12 we can conclude that when all attributes are uncertain, possible consistency is hard, unless the FDs are all trivial (and then all three problems are clearly solvable in polynomial time); this is under the reasonable (and necessary) assumption that F has no *consensus FDs*, that is, the left hand side of every FD is nonempty [21]. We later discuss this assumption. This emphasizes the importance of having a data model that distinguishes between certain and uncertain attributes.

▶ **Theorem 14.** *Let F be a nontrivial set of FDs over a relation schema R where all attributes are uncertain, none being a consensus FD. Then possible consistency is NP-complete.*

The proof selects between a reduction from MPD with the FD $?A \to B$ (Lemma 10) and a reduction from MPD with the matching constraint $?A \leftrightarrow ?B$ (Lemma 11), depending on the structure of F.

We note that the assumption that F has no consensus FDs is necessary. For example, for $F = \{\emptyset \to ?A\}$, which is nontrivial, we can find a most probable database by considering every possible value a for $?A$, computing the probability of selecting a in all distributions, and finally using the value with the maximal probability.

From Theorem 14 we immediately conclude the hardness of the three problems on *every* nontrivial set of FDs in the block-independent-disjoint (BID) model of probabilistic databases [26], due to the translation mentioned in the Introduction.

6 General Sets of Unary Functional Dependencies

In Section 5.1, we studied the complexity of the three problems in the case of a binary schema, and we gave a full classification of the different possible sets of FDs. In this section, we extend these results to a general classification (dichotomy) for every set of *unary* FDs, that is, FDs with a single attribute on the left side. Our result uses a decomposition technique that we devise next.

6.1 Reduction by Decomposition

In this section, we devise a decomposition technique that allows us to reduce our computational problems from one set of FDs into multiple smaller subsets of the set. This technique is stated in the next theorem. After the theorem, we show several consequences that illustrate the use of the technique. Later, we will use these consequences to establish a full classification of complexity for the sets of unary FDs.

▶ **Theorem 15.** *Let F be a set of FDs over a relation schema R. Suppose that $F = F_1 \cup F_2$ and that all attributes in $\mathbf{Att}(F_1) \cap \mathbf{Att}(F_2)$ are certain (unmarked). Each of the three problems (in Definition 6) can be solved in polynomial time if its version with F_j and $\mathbf{Att}(F_j)$ is solvable in polynomial time for both $j = 1$ and $j = 2$.*

Proof sketch. Let $U_j = \pi_{\mathbf{Att}(F_j)} U$ for $j = 1, 2$. We show the following:

1. U is possibly consistent w.r.t. F if and only if U_1 and U_2 are possibly consistent w.r.t. F_1 and F_2, respectively.
2. MPDs of U_1 and U_2 can be easily combined to produce an MPD of U.
3. $\Pr_U(F) = \Pr_{U_1}(F_1) \cdot \Pr_{U_2}(F_2)$.

The full details are provided in [10]. ◀

An immediate conclusion from Theorem 15 is that we can eliminate the FDs that involve only certain attributes if we know how to deal with the remaining FDs.

▶ **Corollary 16.** *Let F be a set of FDs over a relation schema R. Let $X \to Y$ be an FD in F, and suppose that all attributes in X and Y are certain. Then each of the three problems (in Definition 6) is polynomial-time reducible to its version with R and $F \setminus \{X \to Y\}$.*

▶ Remark 17. Eliminating the FDs over the certain attributes is not always beneficial, since these FDs might be needed for applying a polynomial-time algorithm. As an example, consider the following set of FDs: $\{?A \to B\,,\, B \to C\,,\, C \to ?A\}$. As we will show later, for this set of FDs we can find a most probable database in polynomial time. However, we will also show that possible consistency is NP-hard for the subset $\{?A \to B\,,\, C \to ?A\}$. Hence, $B \to C$ is needed for the polynomial-time algorithm. ⌟

The following consequence of Theorem 15 identifies a general tractable case: the problems are solvable in polynomial time if uncertain attributes do not appear in the left side of the FDs (but they can appear in the right side or outside of the FDs).

▶ **Theorem 18.** *Let F be a set of FDs. If the left side of every FD includes only certain attributes, then each of the three problems (in Definition 6) is solvable in polynomial time.*

Proof sketch. Assume, without loss of generality, that each FD in F contains a single attribute on the right side. For every $A \in \mathbf{Att}(F)$, let F_A be the subset of F that contains all FDs with A being the right side (i.e., all FDs of the form $X \to A$). Then $F = \cup_{A \in \mathbf{Att}(F)} F_A$.

Note that sets F_A and F_B, where $A \neq B$, share only certain attributes. This is true since our assumption implies that an uncertain attribute $?A$ can appear only in $F_{?A}$. Hence, we can apply Theorem 15 repeatedly and conclude that we need a polynomial-time solution for each $F_{?A}$. In the full version [10], we show that we can obtain that using a similar concept to the algorithm for $A \rightarrow ?B$ from Section 5.1.1. ◀

6.2 Classification

We now state the precise classification of the complexity of the problems in the case of unary FDs. The statement uses the following terminology. Let F be a set of unary FDs. Recall that two attributes A and B and are *equivalent* if they have the same closure, that is, $\{A\}^+_F = \{B\}^+_F$. An attribute A is called a *sink* if $\{A\}^+_F = \{A\}$, that is, A does not appear in the left hand side of any nontrivial FD. In this section, we will prove the following classification (trichotomy) result, which is also illustrated in Figure 3.

▶ **Theorem 19.** *Let F be a set of unary FDs over a relation schema R. Then following hold.*

1. *If every uncertain attribute is either a sink or equivalent to a certain attribute, then a most probable database can be found in polynomial time; otherwise, possible consistency is NP-complete.*
2. *If every uncertain attribute is a sink, then the probability of consistency can be calculated in polynomial time; otherwise, it is* $\mathrm{FP}^{\#\mathrm{P}}$*-complete.*

The following examples illustrate the instantiation of the theorem to specific scenarios.

▶ **Example 20.** We give several examples for the case of a ternary schema $\{A, B, C\}$. Consider the following sets of FDs:

$$F_1 := \{A \rightarrow B \rightarrow ?C\} \qquad F_2 := \{A \rightarrow ?B \rightarrow C\} \qquad F_3 := \{A \leftrightarrow ?B \rightarrow C\}$$

Theorem 19 tells us the following. All three problems are solvable in polynomial time in the case of F_1, since the uncertain attribute $?C$ is a sink. In the case of F_2, we can see that $?B$ is neither a sink nor equivalent to any certain attribute; hence, all three problems are intractable for F_2. In the case of F_3, the attribute $?B$ is not a sink but is equivalent to the certain attribute A. Hence, the probability of consistency for F_3 is $\mathrm{FP}^{\#\mathrm{P}}$-complete, but we can find a most probable database in polynomial time. ⌟

Next, we illustrate Theorem 19 on our running example.

▶ **Example 21.** Consider again the CIR U_2 of Figure 2. Consider the following constraints.

1. business $\rightarrow$?spokesperson?location
2. ?spokesperson $\rightarrow$?location
3. business $\leftrightarrow$?spokesperson $\rightarrow$?location

For the first constraint, all three problems are tractable since both ?spokesperson and ?location are sinks. For the second constraint, all three problems are intractable since ?spokesperson is neither a sink nor equivalent to any certain attribute. For the third constraint, a most probable database can be found in polynomial time since ?spokesperson is equivalent to the certain business and ?location is a sink, but the probability of consistency is $\mathrm{FP}^{\#\mathrm{P}}$-complete since ?spokesperson is not a sink. ⌟

In the remainder of this section, we prove each of the two parts of Theorem 19 separately.

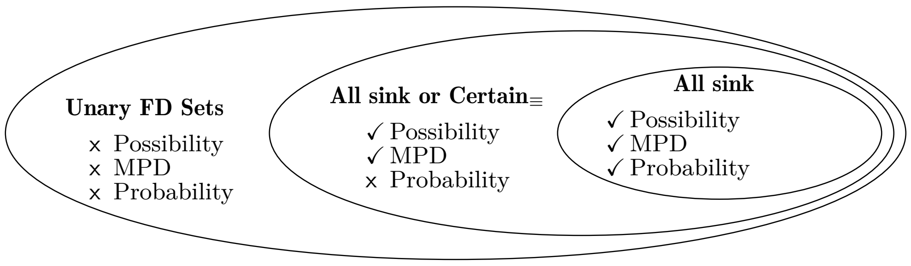

Figure 3 Classification of the complexity of consistency problems for sets of unary FDs. (See Table 1 for the naming of the problems.) "All sink" refers to the case where every uncertain attribute is sink, and "All sink or Certain$_\equiv$" refers to the case where every uncertain attribute is either a sink or equivalent to a certain attribute.

6.2.1 Part 1 of Theorem 19 (Possible Consistency and MPD)

We first prove the tractability side of Part 1 of the theorem.

▶ **Lemma 22.** *Let F be a set of unary FDs over a schema R. If every uncertain attribute is either a sink or equivalent to a certain attribute, then a most probable database can be found in polynomial time.*

Proof sketch. The idea is to define a set $F_{?A}$ of FDs for every uncertain attribute $?A \in ?\mathbf{Att}(U)$, and a set F' of FDs where all left-side attributes are certain, such that F is equivalent to $F' \cup \bigcup_{?A \in ?\mathbf{Att}(U)} F_{?A}$. Then, we repeatedly apply Theorem 15 to reduce the original problem to instances that are solvable in polynomial time by Proposition 8 and Theorem 18. ◀

For the hardness side of Part 1 of Theorem 19, we will need the following lemma, which generalizes the case of $?A \leftrightarrow ?B$ from Lemma 11.

▶ **Lemma 23.** *Let $R = \{?A_1, \dots, ?A_k\}$ consist of $k > 1$ uncertain attributes, and suppose that F is a set of FDs stating that all attributes in R are equivalent. Then possible consistency is NP-complete.*

The next lemma states the hardness side of Part 1 of Theorem 19.

▶ **Lemma 24.** *Let F be a set of unary FDs over a schema R. If there is an uncertain attribute that is neither a sink nor equivalent to a certain attribute, then possible consistency is NP-complete.*

Proof sketch. Let $?A$ be an attribute that is neither a sink nor equivalent to a certain attribute. Let X be the closure of $?A$ and X' be $X \setminus \{?A\}$. Observe the following. First, X' must be nonempty since $?A$ is not a sink. Second, if any attribute in X' implies $?A$ then it is equivalent to $?A$, and then it is necessarily uncertain. We consider two cases:

1. No attribute in X' implies $?A$.
2. Some attribute in X' implies $?A$.

For the first case, we show a reduction from $?A \to B$, where possible consistency is NP-complete due to Lemma 10. For the second case, let $?B_1, \dots, ?B_\ell$ be the set of all attributes in X' that imply $?A$. As said above, each $?B_j$ must be uncertain. Then all of $?B_1, \dots, ?B_\ell, ?A$ are equivalent. We show a reduction from the problem of Lemma 23 where $k = \ell + 1$. ◀

6.2.2 Part 2 of Theorem 19 (Probability of Consistency)

We now move on to Part 2. The tractability side follows immediately from Theorem 18, since if all uncertain attributes are sinks, then all left-side attributes are certain (up to trivial FDs $?A \rightarrow ?A$ that can be ignored). Hence, it remains to prove the hardness side of Part 2 of Theorem 19. We start with the following lemma, where we use a reduction from the case of $A \leftrightarrow ?B$, where probability of consistency is $\mathrm{FP}^{\#\mathrm{P}}$-complete by Proposition 9, to establish hardness for a more general case.

▶ **Lemma 25.** *Let F be a set of unary FDs over a schema R. If at least one uncertain attribute is equivalent to a certain attribute, then the probability of consistency is $\mathrm{FP}^{\#\mathrm{P}}$-complete.*

We can now complete the proof of the hardness side of Part 2.

▶ **Lemma 26.** *Let F be a set of unary FDs over a schema R. If there is at least one uncertain attribute that is not a sink, then the probability of consistency is $\mathrm{FP}^{\#\mathrm{P}}$-complete.*

Proof sketch. Let $?A$ be an uncertain attribute that is not a sink. Let $Y = (?A)^+_F \setminus \{?A\}$. Note that Y is nonempty, since $?A$ is not a sink. If any attribute B in Y functionally determines $?A$, then we can use this attribute as a certain attribute (even if it is uncertain) and use Lemma 25, since $?A$ is equivalent to B. Otherwise, suppose that no attribute in Y determines $?A$. For this case, we show (in [10]) a reduction from $?A \rightarrow B$, where the probability of consistency is $\mathrm{FP}^{\#\mathrm{P}}$-hard according to Lemma 10. ◀

6.2.3 Recap

We can now complete the proof of Theorem 19. For Part 1, the tractability side is given by Lemma 22, and the hardness is given by Lemma 24. As for Part 2, the tractability side follows immediately from Theorem 18, and the hardness side is stated in Lemma 26.

7 Conclusions

We defined the concept of a CIR and studied the complexity of three problems that relate to consistency under FDs: possible consistency, finding a most probable database, and the probability of consistency. A seemingly minor feature of the definition of a CIR is the distinction between certain and uncertain attributes; yet, this distinction turns out to be crucial for detecting tractable cases. We gave classification results for several classes of FD sets, including a single FD, a matching constraint, and arbitrary sets of unary FDs. We also showed that if all attributes are allowed to be uncertain, then the first two problems are intractable for every nontrivial set of FDs.

This work leaves many problems for future investigation. Within the model, we have not yet completed the classification for the whole class of FD sets, where the problem remains open. Recall that a full classification is known for the most probable database for tuple-independent databases [21]. Moreover, as we hit hardness already for simple cases (e.g., $?A \rightarrow B$), it is important to identify realistic properties of the CIR that reduce the complexity of the problems and allow for efficient algorithms.

Going beyond the framework of this paper, we plan to study additional types of constraints that are relevant to data cleaning [9], such as conditional FDs, denial constraints, and foreign-key constraints (where significant progress has been recently made in the problem of consistent query answering [14]). Another useful direction is to consider *soft* or *approximate* versions of the constraints, where it suffices to be consistent to some quantitative extent [5,16,20].

Finally, we have made the assumption of probabilistic independence among the cells as this is the most basic setting to initiate this research. To capture realistic correlations in the database noise, it is important to extend this work to data models that allow for (learnable) probabilistic dependencies, such as Markov Logic [17].

References

1 Serge Abiteboul, T.-H. Hubert Chan, Evgeny Kharlamov, Werner Nutt, and Pierre Senellart. Aggregate queries for discrete and continuous probabilistic XML. In *ICDT*, ACM International Conference Proceeding Series, pages 50–61. ACM, 2010.

2 Periklis Andritsos, Ariel Fuxman, and Renée J. Miller. Clean answers over dirty databases: A probabilistic approach. In *ICDE*, page 30, 2006.

3 Felix Bießmann, Tammo Rukat, Philipp Schmidt, Prathik Naidu, Sebastian Schelter, Andrey Taptunov, Dustin Lange, and David Salinas. Datawig: Missing value imputation for tables. *J. Mach. Learn. Res.*, 20:175:1–175:6, 2019.

4 Marco Calautti, Ester Livshits, Andreas Pieris, and Markus Schneider. Counting database repairs entailing a query: The case of functional dependencies. In *PODS*, pages 403–412. ACM, 2022.

5 Nofar Carmeli, Martin Grohe, Benny Kimelfeld, Ester Livshits, and Muhammad Tibi. Database repairing with soft functional dependencies. In *ICDT*, volume 186 of *LIPIcs*, pages 16:1–16:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

6 Reynold Cheng, Jinchuan Chen, and Xike Xie. Cleaning uncertain data with quality guarantees. *Proc. VLDB Endow.*, 1(1):722–735, 2008.

7 Sara Cohen, Benny Kimelfeld, and Yehoshua Sagiv. Incorporating constraints in probabilistic XML. *ACM Trans. Database Syst.*, 34(3):18:1–18:45, 2009.

8 Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875. Morgan Kaufmann, 2004.

9 Wenfei Fan and Floris Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

10 Amir Gilad, Aviram Imber, and Benny Kimelfeld. The consistency of probabilistic databases with independent cells, 2022. `doi:10.48550/arXiv.2212.12104`.

11 Erich Grädel, Yuri Gurevich, and Colin Hirsch. The complexity of query reliability. In *PODS*, pages 227–234. ACM Press, 1998.

12 Eric Gribkoff, Guy Van den Broeck, and Dan Suciu. The most probable database problem. In *BUDA*, 2014. URL: `http://www.sigmod2014.org/buda`.

13 Venkatesan Guruswami. Inapproximability results for set splitting and satisfiability problems with no mixed clauses. In *APPROX*, volume 1913 of *LNCS*, pages 155–166. Springer, 2000.

14 Miika Hannula and Jef Wijsen. A dichotomy in consistent query answering for primary keys and unary foreign keys. In *PODS*, pages 437–449. ACM, 2022.

15 Alireza Heidari, Joshua McGrath, Ihab F. Ilyas, and Theodoros Rekatsinas. HoloDetect: Few-shot learning for error detection. In *SIGMOD Conference*, pages 829–846. ACM, 2019.

16 Abhay Kumar Jha, Vibhor Rastogi, and Dan Suciu. Query evaluation with soft-key constraints. In *PODS*, pages 119–128. ACM, 2008.

17 Abhay Kumar Jha and Dan Suciu. Probabilistic databases with markoviews. *Proc. VLDB Endow.*, 5(11):1160–1171, 2012.

18 Solmaz Kolahi and Laks V. S. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *ICDT*, volume 361 of *ACM*, pages 53–62. ACM, 2009.

19 Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

20 Ester Livshits, Alireza Heidari, Ihab F. Ilyas, and Benny Kimelfeld. Approximate denial constraints. *Proc. VLDB Endow.*, 13(10):1682–1695, 2020.

21 Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. Computing optimal repairs for functional dependencies. *ACM Trans. Database Syst.*, 45(1):4:1–4:46, 2020. doi:10.1145/3360904.

22 Ester Livshits, Benny Kimelfeld, and Jef Wijsen. Counting subset repairs with functional dependencies. *J. Comput. Syst. Sci.*, 117:154–164, 2021.

23 Dany Maslowski and Jef Wijsen. A dichotomy in the complexity of counting database repairs. *J. Comput. Syst. Sci.*, 79(6):958–983, 2013.

24 Chris Mayfield, Jennifer Neville, and Sunil Prabhakar. ERACER: a database approach for statistical inference and data cleaning. In *SIGMOD Conference*, pages 75–86. ACM, 2010.

25 Luyi Mo, Reynold Cheng, Xiang Li, David W. Cheung, and Xuan S. Yang. Cleaning uncertain data for top-k queries. In *ICDE*, pages 134–145, 2013.

26 Christopher Ré and Dan Suciu. Materialized views in probabilistic databases for information exchange and query optimization. In *Proc. VLDB Endow.*, pages 51–62, 2007.

27 Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. HoloClean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.

28 Christopher De Sa, Ihab F. Ilyas, Benny Kimelfeld, Christopher Ré, and Theodoros Rekatsinas. A formal framework for probabilistic unclean databases. In *ICDT*, volume 127 of *LIPIcs*, pages 6:1–6:18, 2019.

29 Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

30 Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

31 Frank S.C. Tseng, Wei-Pang Yang, and Arbee L.P. Chen. Finding a complete matching with the maximum product on weighted bipartite graphs. *Computers & Mathematics with Applications*, 25(5):65–71, 1993.

32 Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

33 Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *STOC*, pages 137–146. ACM, 1982.

34 Jef Wijsen. Database repairing using updates. *ACM Trans. Database Syst.*, 30(3):722–768, 2005.