An epistemic approach to model uncertainty in data-graphs

Abriola Sergio[1,2], Cifuentes Santiago[2], Martinez Maria Vanina[1,2,4], Pardal Nina[1,2], and Pin Edwin[1,3]

[1]Instituto de Ciencias de la Computacion (UBA-CONICET)
[2]Departamento de Computación, Universidad de Buenos Aires
[3]Departamento de Matemática, Universidad de Buenos Aires
[4]Artificial Intelligence Research Institute (IIIA-CSIC)

Abstract

Graph databases are becoming widely successful as data models that allow to effectively represent and process complex relationships among various types of data. *Data-graphs* are particular types of graph databases whose representation allows both data values in the paths and in the nodes to be treated as first class citizens by the query language. As with any other type of data repository, data-graphs may suffer from errors and discrepancies with respect to the real-world data they intend to represent. In this work, we explore the notion of *probabilistic unclean data-graphs*, in order to capture the idea that the observed (unclean) data-graph is actually the noisy version of a clean one that correctly models the world but that we know only partially. As the factors that lead to such a state of affairs may be many, e.g, all different types of clerical errors or unintended transformations of the data, and depend heavily on the application domain, we assume an epistemic probabilistic model that describes the distribution over all possible ways in which the clean (uncertain) data-graph could have been polluted. Based on this model we define two computational problems: *data cleaning* and *probabilistic query answering* and study for both of them their corresponding complexity when considering that the polluting transformation of the data-graph can be caused by either removing (subset), adding (superset), or modifying (update) nodes and edges. For data cleaning, we explore restricted versions when the transformation only involves updating data-values on the nodes. Finally, we look at some implications of incorporating hard and soft constraints to our framework.

Keywords: *Data-graphs, Consistent query answering, Probabilistic query answering, Constraints, Inconsistent databases, Repairing*

1 Introduction

There is an increasing interest in graph databases as a mean to adequately handle both the topology of the data and the data itself, which is specially useful for applications that involve analysis over linked and semi-structured data [5, 9, 24]. Graphs have long been used as medium of representation for a wide range of problems, not only in artificial intelligence and knowledge representation and reasoning, but also in other disciplines beyond computer science. Through a quite simple data structure consisting of nodes and edges, they provide a nice representation alternative with interesting operational properties for flexibly expressing relations, not necessarily ordered or sequentially structured. Though it is possible to represent

such structures within classical relational databases, the resulting representations are not necessarily flexible nor adequate for the type of uses that these repositories have [8]. In these settings, the structure of the database is queried through navigational languages such as *regular path queries* or RPQs [10] that can capture pairs of nodes connected by a specific kind of path. Note that though relational database query languages can capture such expressions through joins on tables, it becomes quite cumbersome to write for non-technical users, and furthermore, the execution of such queries over a considerable large corpus of data can become prohibitive in practice. More expressive query languages have been defined with a tradeoff on evaluation complexity, nevertheless, RPQs and its most common extensions (C2RPQs and NREs [13]) do not interact with data values in the nodes of the graph. For this reason, query languages have been defined for the case of data-graphs (i.e. graph databases where data lies both in the paths and in the nodes themselves), such as REMs and Reg-GXPath [37].

Data-graphs are interesting from an application point of view as they can be used to model real-world knowledge as such encountered in social networks and knowledge graphs, among others. They provide a simple but rich representation of both edges and nodes, and tools for updating and extracting complex data from them, such as identifying especial patterns of interest in such networks like centrality measures [20], communities [38, 25], or even anomalous behavior [34]. It is easy to see that some of the techniques studied in this work can be used in the implementation of particular tasks involved in the maintenance and update of such knowledge networks[26, 32], such as cleaning, link completion, etc.

Besides defining data models and query languages that are adequate for specific application domains, when accessing or querying a data repository we expect to obtain data that comply, at least to a certain extent, with the semantics of the domain, either in terms of quality or by satisfying a series of integrity constraints. This is a major challenge for any non-trivial data-driven application. Specifically for data-graphs, integrity constraints can be expressed in graph databases through *path constraints* [2, 17].

In the literature, two main approaches have been thoroughly explored to handle inconsistency. On the one hand, data cleaning, or data repairing, focuses on frameworks that allow to identify inconsistencies caused by incorrect, missing, and duplicated data, and to restore it to a *clean* state, i.e., a state that satisfies the imposed constraints. On the other hand, repairing the data may not always be the best option, or it may actually be impossible, e.g, we may not have the permission to actually change it. In these cases, a different approach is to develop the means (theory and algorithms) to obtain consistent answers from inconsistent databases without changing the data itself. The field of consistent query answering (CQA), first defined for relational databases [7, 36] has lately been applied to semi-structured data such as graph databases [11].

One of the main drawbacks of traditional approaches to repairing and consistent query answering is that they do not allow to represent the fact that, in general, when we observe the *unclean* data we may not know exactly how the data was corrupted, as a variety of factors may have been involved. In the context of data-graphs, a possible method to represent our uncertain knowledge is to take a Bayesian epistemic approach and assume a prior probability distribution over data-graphs, along with a representation of how the original nodes and edges of data-graphs can be corrupted by some application-dependant noisy process. The core idea is that, in this framework, the observation of a particular unclean data-graph can be lifted via the chosen epistemic model to a probability distribution over the possible clean data-graphs. For these reasons, inspired by the work in [44, 42], in this paper we propose a probabilistic framework for repairing and querying data-graphs, assuming an epistemic model that describes both a prior on possible data-graphs and a distribution over all possible ways in which the clean (uncertain) data-graph could have been polluted. We adapt the definitions from [44] to the context of data-graphs, especially focusing on complexity aspects, and capitalizing on graph-related logics to define natural constraints and problems. Data-graphs allow for a more natural way to express and tackle problems involving constraints such as path constraints, which are more convenient for graph related applications. We also take advantage of common theoretical devices developed for studying the complexity of reasoning problems related to database repairing.

Overall, we aim to find tractable restrictions of *data cleaning* and *PQA* in which relevant data-graph reasoning tasks can be modelled. In particular, we focus on use cases where data is altered at the node level. In this work, we study probabilistic data cleaning and query answering for data-graphs focusing on three types of transformations that can be applied to the data-graph: by either removing or adding nodes and

edges, and by updating data values in the nodes. As far as we know from the literature review, [44] is the only work that deals with issues of complexity for a probabilistic framework such as the one presented in this paper. The adaptation of the framework to data-graphs that we propose brings new challenges related to the management of data in both nodes and edges, and the consequences of using navigational graph logics to the purpose of dealing with inconsistency.

The specific contributions of this work are as follows:

- We define the notion of a probabilistic unclean data-graph model (PUDG) based on the observed data-graph and an epistemic model of the data-graph (EMDG); the latter is composed of a probabilistic distribution over all possible clean data-graphs and a realization model that represents the noisy process which allows us to 'observe' a potentially distorted version of each of the clean versions.

- We restrict the general EMDG to the cases of subset EMDG and superset EMDG, where we consider realization models that either only delete or only add data in the data-graphs, respectively. These notions relate to two common semantics for repairing databases, namely subset and superset repairs [47]. We also consider the Node-Update EMDG version, where only modifications to data-values in the observed data-graph are allowed.

- We study the problem of *data cleaning*, this is, given a probabilistic unclean data-graph $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$, find the most probable data-graph in $\mathcal{I}$ given an epistemic model of the application domain and an observation. We explore different restrictions of the problem, trying to understand the range of complexity that can be involved when reasoning around these structures.

- We consider alternate ways of defining a system of priorities by adding constraints to the model. We observe that the concepts of hard and soft integrity constraints [7, 11, 47] can be easily incorporated in the framework.

- Finally, we study the problem of *probabilistic query answering* (PQA), which computes the probability of a formula η over $\mathcal{I}$, given a probabilistic unclean data-graph $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$.

This work is organized as follows. First, in Section 2 we discuss related work to provide some background to the proposal. In Section 3 we introduce the necessary preliminaries and notation for the syntax and semantics for our probabilistic model (PUDG). In Section 4 we define the problem of probabilistic data cleaning and study its complexity for Subset, Superset, and Update PUDGs. We finish the section by considering restricted versions of Node-Update PUDG by applying different kinds of cardinality constraints to this framework. In Section 6 we briefly discuss the generalization of our framework to consider soft and hard constraints, and study the complexity of solving the Data Cleaning problem in some particular cases of hard constraints. Later, in Section 7 we study the problem of probabilistic query answering for PUDGs focusing on the subset and superset versions of the epistemic model. Finally, in Section 8 we discuss some final remarks and possible continuation of this work.

2 Related Work

Probabilistic databases have been studied over the last 20 years by both the Database and the Artificial Intelligence communities [50]. These studies were motivated by a variety of applications, such as database repairing [42], modeling uncertain data [45], data cleaning [44] and approximate query processing [52]. Nonetheless, the main purpose of probabilistic databases is to extend today's database technology to handle uncertain data while avoiding developing a new artifact from scratch [50]. Many of the usual database techniques, such as query optimization and indexes, can be carried over (with some necessary adjustments) to a probabilistic database, to follow the new probabilistic semantics of the language.

The main problem that was studied in this context is query evaluation [50, 21, 40]: given a query Q and a probabilistic database D, compute the answer of Q over D. This problem is known as *probabilistic*

query answering or *probabilistic query evaluation* (PQE), which frames the task of finding the probability associated to each answer to Q that is yielded by D. Though several semantics have been defined, overall, the task to compute the probability of Q over D can be reduced to enumerating every possible world d (a non-probabilistic database) that satisfies Q, assigning to that answer a weight that is related to the probability of d given the semantics. Through this interpretation, it can be seen that the PQE problem is similar to the *weighted model counting problem* [14, 31], and it is the case that tight bounds on the complexity of the PQE problem can be deduced using techniques from the latter one. An important dichotomy result of the area is that, for a fixed query Q, PQE is either a $\#P-complete$ problem (as usual for problems related to counting satisfying assignments of formulas) or it can be solved in polynomial time [23].

The probabilistic database D is commonly restricted to be a *tuple-independent database* (TID), in which every tuple is an independent probabilistic event with its own associated value. Even though more expressive alternatives were proposed, such as block-disjoint-independent databases [22] or seeing attributes as random variables [6], TIDs are the best understood alternative to date, and are already being used in applications such as relational embeddings [27]. In some cases, the probabilistic database D is restricted further by requesting it to be *symmetric*: any two tuples of the same relation are conditioned to have the same probability. When considering this special case, the PQE problem can be solved in polynomial time for a larger class of queries [49].

A problem that is quite similar to the *data cleaning* version we study in this work is the problem of computing the *most probable database* (MPD), studied in [30]. In the MPD problem, the task is to find the most probable deterministic database d that satisfies a given query Q over a probabilistic database D. This problem already renders intractable when the probabilistic database D is assumed to be a TID and the query Q a set of key or functional dependencies with a particularly simple structure [30]. The recent work in [29] deals with a version of the most probable database problem over *cell-independent relations* (CIR), a similar structure to TIDs, where the uncertainty relies over the contents of the cells instead of the tuples of the database, and in the presence of FD constraints. Most of the complexity results they obtained for this problem are intractable even for simple classes of FDs such as matching constraints, and arbitrary sets of unary FDs. The classification (i.e., whether the problem is tractable or not) depends on both the combination of FDs in the given set of constraints, and whether the uncertain attributes appear on the left or right hand of the constraints.

Probabilistic models have already been developed in other contexts, beyond the relational one, such as XML [35, 46, 1], ontologies [43], and graphs [36]. In the case of graphs, most of the attention has been placed on the problem of querying a probabilistic graph database [36, 4, 39], where the underlying distribution over the state of the database is defined through edge independent probabilities, in an analogous way as in TIDs. Queries usually consist of some kind of path or graph pattern, and the satisfiability of a query is defined in terms of the probability that the complete pattern can be found in the graph based on the independent probabilities of the edges. As far as we know, there is no work on modelling the graph databases considering data in the nodes (in the same way as the XML trees do) and allowing the graph patterns to interact with them.

3 Definitions

Fix a finite set of edge labels Σ_e and a countable set (either finite or infinite enumerable) of data values Σ_n (sometimes called data labels), which we assume non-empty, and such that $\Sigma_e \cap \Sigma_n = \emptyset$. A (finite) **data-graph** G is a tuple (V_G, L_G, D_G), or just (V, L, D) if G is clear, where V is a finite set of natural numbers, L is a mapping from $V \times V$ to $\mathcal{P}(\Sigma_e)$ defining the edges of the graph, and D is a function mapping the nodes from V to data values in Σ_n. We denote by E_G the set $\{(v, e, w) \mid v, w \in V, e \in L(v, w)\}$, which can be understood as the set of edges of the graph. Also, we denote by $G \subseteq G'$ when $V_G \subseteq V_{G'}$, $E_G \subseteq E_{G'}$, and $D_G(x) = D_{G'}(x)$ for every $x \in V_G$.

See Figure 1 for a visual example of a data-graph.

Definition 1. A **probabilistic data-graph** $\mathcal{I}$ is a probability distribution over a set of data-graphs. More

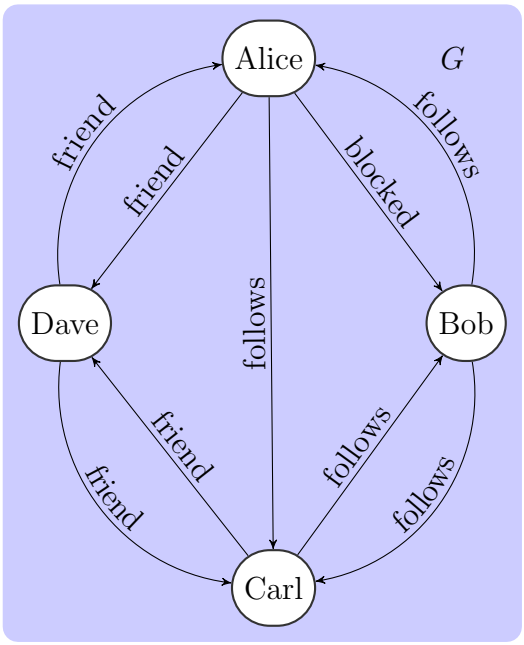


Figure 1: A data-graph G with set of nodes $V = \{1, 2, 3, 4\}$ and data values $D(1) = \text{ALICE}$, $D(2) = \text{DAVE}$, $D(3) = \text{CARL}$ and $D(4) = \text{BOB}$, representing the names of the individuals in a social network, and the relation between any two individuals are either FRIEND, FOLLOWS, or BLOCKED.

precisely, we consider $\mathcal{I} : U \to [0, 1]$, where U is the set of all data-graphs over Σ_e, Σ_n, and

$$\sum_{G \in U} \mathcal{I}(G) = 1.$$

For simplicity, we will sometimes abuse the notation and write $G \in \mathcal{I}$, or G in $\mathcal{I}$, to refer to the case where G is in the support of $\mathcal{I}$ (i.e., when $\mathcal{I}(G) > 0$).

Intuitively, a probabilistic data-graph $\mathcal{I}$ consists of a distribution over all the data-graphs with the same edge-label and data-value sets. The value $\mathcal{I}(G)$ represents our prior domain knowledge as a probability, that is, the probability of G being 'possible', or 'true'. The probability distribution $\mathcal{I}$ can represent different types of uncertainty, such as the details of a concrete database, our priors about the relative frequency of different databases within a certain class, etc.

Example 2. As a toy example, we can consider a probabilistic data-graph that represents our epistemic state about the social relationship between four particular persons.

Consider then $\Sigma_e = \{\text{FRIEND}\}$ and Σ_n a (possibly infinite) set of valid names. Let $V = \{1, 2, 3, 4\}$, where each element of V represents a different person, and U consists of data-graphs over Σ_e, Σ_n, V. Now, assume that we are certain about the name of the four individuals to be $\{a_1, a_2, a_3, a_4\}$. Furthermore, assume that we know that 1 and 4 are mutual friends, but we are completely ignorant about other relationships among the four persons. Then, we could represent our epistemic state as the single probabilistic data-graph $\mathcal{I} : U \to [0, 1]$ where all the following conditions hold:

- $\mathcal{I}(G) = 0$ for any G such that, for some $1 \leq i \leq 4$, $D(i) \neq a_i$.
- $\mathcal{I}(G) = 0$ for any G such that either $(1, \text{FRIENDS}, 4) \notin E_G$ or $(4, \text{FRIEND}, 1) \notin E_G$.
- $\mathcal{I}(G) = 0$ for any G such that $(i, \text{FRIEND}, i) \in E_G$ for some i.
- $\mathcal{I}$ assigns the same probability to any G that does not satisfy the previous conditions.

The first condition ensures that the names of the individuals are precisely those that we know for certain. The second condition enforces that 1 and 4 are mutual friends, as requested. In other words, any graph with positive probability needs to have an edge from 1 to 4 with label FRIEND and analogously from 4 to 1. As for the third condition, we ask that no individual is a friend of itself, this is, that every data-graph that represents this social relationship has no loops. Finally, the last condition states that those graphs that satisfy the first three constraints are equiprobable, disregarding of which other relationships they model.

In particular, it is easy to see that there are exactly $2^{2*3+2*2}$ data-graphs that fulfil these conditions, and thus they have assigned a non-negative probability, which is precisely $\frac{1}{1024}$. See Figure 2 for an illustration.

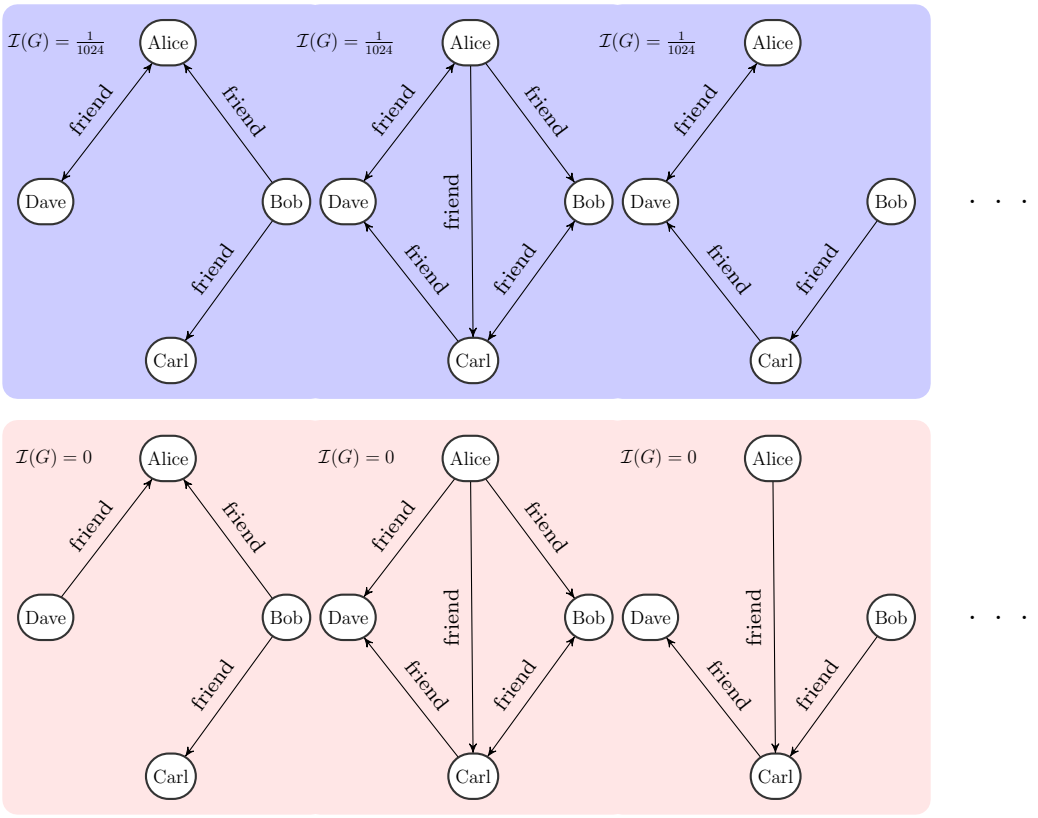


Figure 2: A toy example of a probabilistic data-graph, where nodes $1, 2, 3, 4$ are labeled with the Σ_n data values $a_1 = \text{ALICE}, a_2 = \text{BOB}, a_3 = \text{CARL}$ and $a_4 = \text{DAVE}$, respectively.

Having defined a probabilistic data-graph, we now present the notion of transformation among possible states of a data-graph, via a probabilistic criterion.

Definition 3. Given by Σ_e and Σ_n, let U be the universe of data-graphs and P be the universe of probabilistic data-graphs. A **realization model** (or a **noisy observer**) is a function $\mathcal{R} : U \to P$. That is, $\mathcal{R}$ maps data-graphs $G \in U$ into probabilistic data-graphs.

Intuitively, $\mathcal{R}$ allows us to 'observe' a potentially distorted version of a clean data-graph. Figure 3 shows a toy example of a social network illustrating these concepts.

We now formalize the Epistemic Model of a Data-Graph, which incorporates both our prior domain knowledge about the probabilities of different data-graphs and our understanding on how our noisy observations can differ from the underlying reality.

Definition 4 (EMDG)**.** Let U be the universe of data-graphs given by Σ_e and Σ_n. We define an **Epistemic Model of a Data-graph (EMDG)** as a pair $(\mathcal{I}, \mathcal{R})$, where $\mathcal{I}$ is a probabilistic data-graph over U, and $\mathcal{R}$ is a noisy observer.

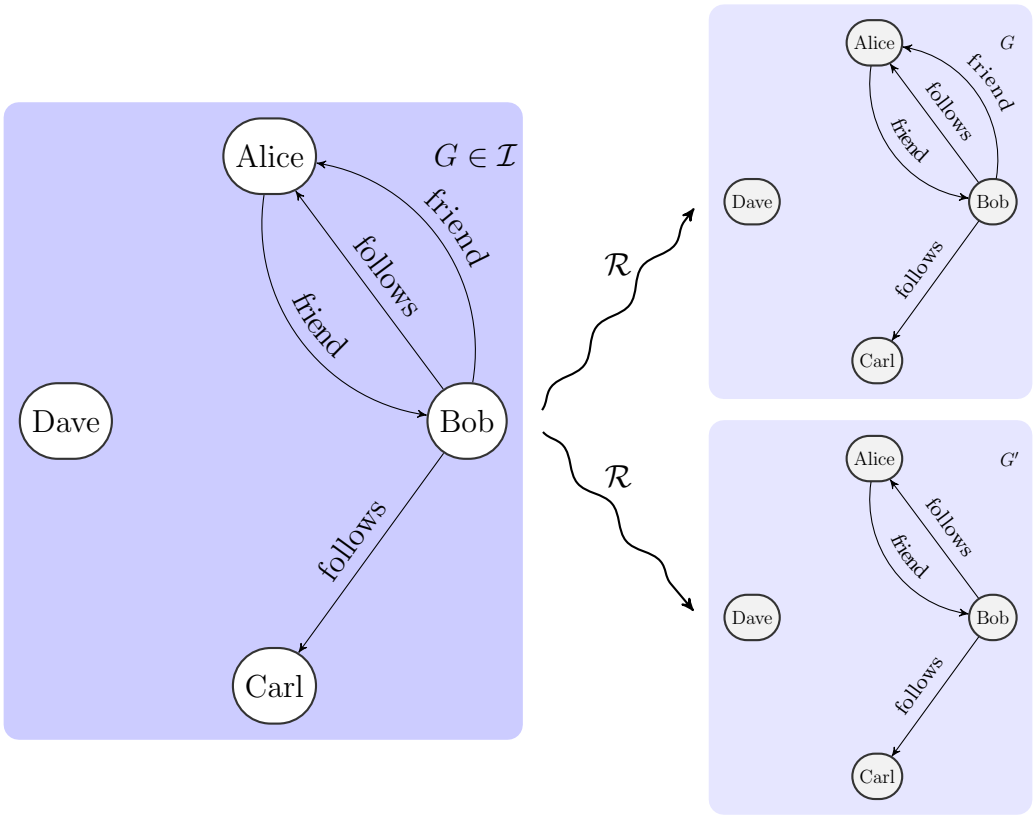


Figure 3: Two possible actions of a particular noisy observer $\mathcal{R}$ over a given data-graph G in $\mathcal{I}$. In one case, with probability $\mathcal{R}(G)(G)$, the original database is left unchanged. In the other case, with probability $\mathcal{R}(G)(G')$, one particular edge of the original database is deleted.

Another possible interpretation for EMDGs is that $\mathcal{I}$ represents idealized databases over certain domain, and $\mathcal{R}$ transforms each idealized version into many possible error-prone implementations. For example, $\mathcal{I}$ could represent bibliographical databases, and $\mathcal{R}$ could produce databases where some type of data-duplication errors have been made.

Inspired by [42, 44], we define a probabilistic unclean data-graph in the following way:

Definition 5 (PUDG)**.** We define a **Probabilistic Unclean Data-graph (PUDG)** as a triple $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$, where:

- $(\mathcal{I}, \mathcal{R})$ is an EMDG.
- G' is the observed or unclean data-graph.

Intuitively, G' is the current observable state of the database, which we suspect incorrect because of errors introduced due to data entry mistakes, data integration problems, or other kind of operations that could have polluted the original (unknown) data-graph G. We have some knowledge about the original state of the data-graph prior to these issues, which is modeled by the distribution $\mathcal{I}$; and we also have some idea of the possible ways these original data-graphs from $\mathcal{I}$ could have become unclean, which is modeled by $\mathcal{R}$.

Given G' and $\mathcal{R}$, we define the **cosupport of G'** as the set $\{G \in U \mid G' \in \mathcal{R}(G)\}$. We denote its cardinality by $\sigma_{\mathcal{R}}(G')$, or just $\sigma(G')$ if $\mathcal{R}$ is clear from the context. Throughout this paper, we will always consider EMDGs having finite cosupport for every G'.

Using the intuition of Bayes' theorem, given an EMDG $(\mathcal{I}, \mathcal{R})$, we now define a function whose domain is U and whose image for each data-graph G' is a particular probabilistic data-graph, such that the function returns the probability that the data-graph G was the clean data-graph transformed via the noisy observer

$\mathcal{R}$ into the observed data-graph G' when taking into account the probabilities defined by $\mathcal{I}$. More formally, we denote said function by $\boldsymbol{\mathcal{B}_{\mathcal{I},\mathcal{R}}}$, and define it as:

$$\mathcal{B}_{\mathcal{I},\mathcal{R}}(G')(G) \stackrel{\text{def}}{=} \frac{\mathcal{I}(G) \times \mathcal{R}(G)(G')}{\sum_{H \in \mathcal{I}}(\mathcal{I}(H) \times \mathcal{R}(H)(G'))} \tag{1}$$

whenever there exists $H \in \mathcal{I}$ such that $G' \in \mathcal{R}(H)$, and 0 otherwise. Notice that in the former case, (1) is well-defined as a probabilistic data-graph given that the cosupport of G' is finite. Here, given the observed data-graph G' and a data-graph G, $\mathcal{R}(G)(G')$ is the probability that G' is the graph that results from transforming G through $\mathcal{R}$. Note that this does not take into account the prior given by $\mathcal{I}(G)$. The denominator is the normalization factor over all data-graphs with positive probability in $\mathcal{I}$ that could lead to G' through $\mathcal{R}$.

Example 6. Continuing with our running example on social networks, we can ask whether everyone in a network is connected, in any direction, via at most 3 steps of an edge FRIEND or FOLLOWS.

For the PUDG $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$, the probability that the property holds corresponds to the answer of the `Global PQA` problem for the path expression $\alpha = (\text{FRIEND} \cup \text{FOLLOWS})^{0,3}$. Figure 4 illustrates a particular case. Here we have that the resulting output would be 0.1:

$$\begin{gathered}
\mathcal{B}(G')(G_1) = 0.5 \text{ and } \alpha \text{ does } \textit{not} \text{ hold over all pairs in } G_1 \\
\mathcal{B}(G')(G_2) = 0.4 \text{ and } \alpha \text{ does } \textit{not} \text{ hold over all pairs in } G_2 \\
\mathcal{B}(G')(G_3) = 0.1 \text{ and } \alpha \textit{ does} \text{ hold over all pairs in } G_3 \\
\Longrightarrow \\
\texttt{Global PQA}_\alpha((\mathcal{I}, \mathcal{R}, G')) = 0.1
\end{gathered}$$

On the other hand, the result for the path expression $\overline{\alpha}$ would be 0.9.

Remark 7. Intuitively, $\mathcal{B}_{\mathcal{I},\mathcal{R}}(G')(G)$ is the conditional probability of G given the observation of G', taking into account the distribution over possible states $\mathcal{I}$ and the realization model $\mathcal{R}$. We may write $\mathcal{B}$ without the subscript when both $\mathcal{I}$ and $\mathcal{R}$ are clear from the context. Even though we defined $\mathcal{B}(G')$ as 0 when the cosupport of G' intersected with the support of $\mathcal{I}$ is empty, in the proofs and examples we mostly consider and argue about the (much more interesting) non-empty case, so it will be common to refer to $\mathcal{B}(G')$ as a probabilistic data-graph regardless of the observed data-graph G'.

As it is usually the case, we are going to assume that $\mathcal{I}$, $\mathcal{R}$, and $\mathcal{B}$ are all computable functions over adequate representations of their domains and codomains. When proving complexity bounds for problems related to EMDGs (or rather PUDGs, defined later in this work) we will be concerned with the set of probabilistic databases and realization models that can be computed efficiently. This means that, given G, we consider $\mathcal{I}$'s and $\mathcal{R}$'s such that we can compute $\mathcal{I}(G)$ and $\mathcal{R}(G)$ in time $poly(|G|)$. We also ask for the distribution $\mathcal{B}(G')$ to be computable in polynomial time given its input and output: that is, we can compute it in $poly(|G'| + |\mathcal{B}(G')|)$. Note that defining an arbitrary efficiently computable $\mathcal{I}$ and $\mathcal{R}$ does not imply that $\mathcal{B}(G')(G)$ can be computed in polynomial time.

Example 8. A particular example of the previous possible interpretation of an EMDG is shown in Figure 3, where a social network is incompletely observed via the 'deletion' of a single edge. Here, the data values (in Σ_n) are name identifiers ALICE, BOB, CARL, DAVE, and the possible edge-labels (in Σ_e) are FRIEND and FOLLOWS. In this case, one possible action of $\mathcal{R}$ over the data-graph G is leaving it unchanged, while the other shown possibility is the deletion of a FRIEND edge in the direction from Bob to Alice. Assuming that the representation in the figure is complete, we have that $\mathcal{R}(G)(G) + \mathcal{R}(G)(G') = 1$, since $\mathcal{R}(G)$ is a probabilistic data-graph.

Example 9. Let $\Sigma_e = \{\text{FRIEND\_OF}\}$, Σ_n be a set of common names. Consider S_N the set of finite data-graphs over Σ_e, Σ_n that have at least 2 nodes and no more than N nodes. Let $\mathcal{I} : U \to [0, 1]$ be a uniform

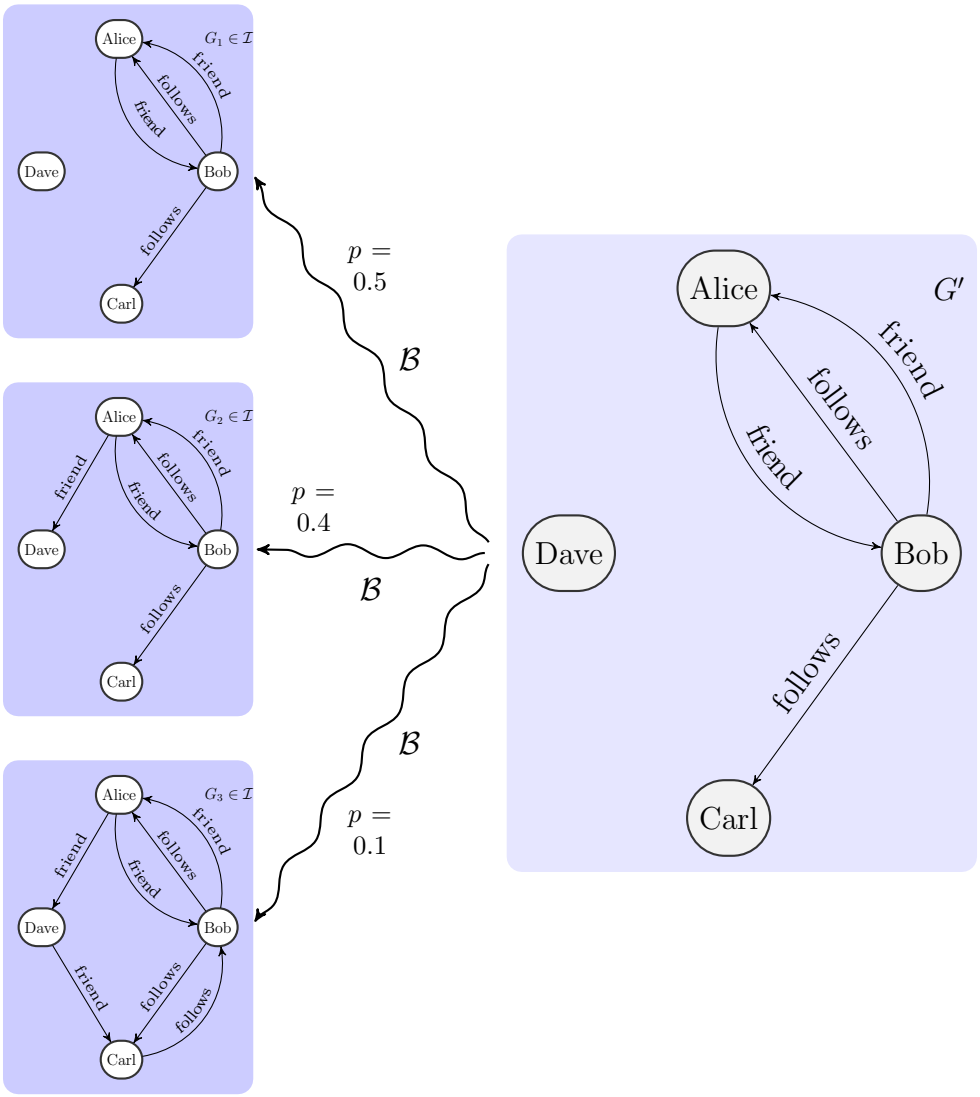


Figure 4: All possible data-graphs with non-zero probability which can be transformed with different non-zero probabilities by $\mathcal{R}$ into G'.

distribution of probabilities among all graphs of S_N, having $\mathcal{I}(G) = 0$ if $G \in U \backslash S_N$, and otherwise with $\mathcal{I}(G)$ given by some prior over the typical shape of social networks.

We can use $\mathcal{R}$ to model the probability of erroneously removing a friendship relation between two people. For example, it could be the case that the probability of this error is a fixed number p between 0 and 1, and that the probability of making this error is independent for each edge; therefore, larger databases would usually be observed with more errors. We can represent this idea by defining $\mathcal{R}$ such that, for any unclean observed state G' and data-graph G, we have:

$$\mathcal{R}(G)(G') = \begin{cases} p^{|E_G \setminus E_{G'}|} \times (1-p)^{|E_{G'}|} & \text{if } G' \subseteq G \text{ and } V_G = V'_G \\ 0 & \text{otherwise} \end{cases}$$

Note that $\mathcal{R}$, as defined above, is indeed a realization model. That is, for every G, $\mathcal{R}(G)$ is a probabilistic data graph:

$$\sum_{G' \in U} \mathcal{R}(G)(G') = \sum_{G' \subseteq G, V_G = V_{G'}} p^{|E_G \setminus E_{G'}|} \times (1-p)^{|E_{G'}|} = 1$$

Furthermore, note that this way of constructing $\mathcal{R}$ can be generalized naturally to larger sets of edge labels Σ_e by assigning different values p_{A} to each $\mathrm{A} \in \Sigma_e$.

Example 10. Let $\Sigma_e = \{\text{AUTHOR\_OF}, \text{NAME}, \text{TITLE}\}$, and let Σ_n be the set of strings over the English alphabet. We can consider an EMDG $(\mathcal{I}, \mathcal{R})$, where $\mathcal{I}$ assigns non-zero probability to all data-graphs that form a viable representation of bibliographical data (captured using some path or tree expression), and where $\mathcal{R}$ can introduce errors by adding outgoing AUTHOR_OF edges from any node that has no incoming edges. For example, $\mathcal{R}$ could erroneously assign co-authorship of the same book, or it could (in a more blatant error) introduce authorship between a person and a name, or between a person and another person.

If we have a PUDG $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$, then the introduction of errors might cause the observed G' not to coincide with the initial clean data-graph. Nonetheless, the nature of this $\mathcal{R}$ allows us to know that the true answer must be a subgraph of G' where the only possible changes, if any, are the removal of AUTHOR_OF edges. Furthermore, the information on $\mathcal{I}$ can also be used; for example, by its own nature we know with certainty that any AUTHOR_OF edges between different authors are errors introduced by $\mathcal{R}$.

In principle, $\mathcal{R}$ might represent any possible transformation between data-graphs. In order to restrict the generality of $\mathcal{R}$ while still preserving a good amount of expressiveness, that remains useful for real-world applications, we will consider realization models that either only delete, only add, or only modify data in the data-graphs (either nodes or edges or data). Considering any of these restrictions is a usual practice when reasoning over uncertainty or inconsistencies. Furthermore, all three notions can be related to common semantics for repairing databases [3, 11, 19, 48].

In order to formalize these ideas, we start by defining different types of EMDG that characterize the different realization models we seek to capture.

Definition 11. A **Subset EMDG** is a pair $(\mathcal{I}, \mathcal{R})$ such that the noisy observer $\mathcal{R}$ satisfies that $G' \subseteq G$ for every data-graph G' such that $\mathcal{R}(G)(G') > 0$.

In a way, we can think that the noisy observer of a Subset EMDG introduces node or edge deletions. To formalize the second notion, we define Superset EMDG in an analogous way:

Definition 12. A **Superset EMDG** is a pair $(\mathcal{I}, \mathcal{R})$ such that the noisy observer $\mathcal{R}$ satisfies that $G' \supseteq G$ for every data-graph G' such that $\mathcal{R}(G)(G') > 0$.

In other words, in this case the noisy observer can add nodes or edges to the original clean graph.

For the case where data modifications are allowed, we provide now formal definitions for two different kinds of data modifications.

Definition 13. We say that $(\mathcal{I}, \mathcal{R})$ is a **Node-Update EMDG** when $\mathcal{R}$ is such that if $\mathcal{R}(G)(G') > 0$, then $V_G = V_{G'}$ and $L_G(v, w) = L_{G'}(v, w)$ for all $u, v \in V_G$.

On the other hand, we say that $(\mathcal{I}, \mathcal{R})$ is an **Update EMDG** when $\mathcal{R}$ is such that if $\mathcal{R}(G)(G') > 0$, then $V_G = V_{G'}$ and $|L_G(v, w)| = |L_{G'}(v, w)|$ for all $u, v \in V_G$.

Intuitively, a node-update only modifies the data-values in the original data-graph, leaving all other structure intact, that is, only the function D might change (recall that D is the function that maps the nodes from V to data values in Σ_n). Figure 6a shows a toy example of a social network where the names of the individuals can be altered by the noisy observer $\mathcal{R}$, while leaving the rest of the network's structure intact. Compare with Figure 3, and more generally with Subset and Superset EMDGs, where the observer is able to modify the data-graph via the removal or addition of edges, but without changing the data in the nodes. Analogously, we may understand an Update EMDG as a transformation that is allowed to modify both the data-values from the nodes and the labels from the edges, while leaving intact the remaining structure (including the number of edges). Figure 6b presents a simple example where more changes are possible than in Figure 6a.

Definition 14 (Π PUDG)**.** A **Π PUDG** is a triple $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$ where $(\mathcal{I}, \mathcal{R})$ is a Π EMDG where $\Pi \in \{\text{Subset}, \text{Superset}, \text{Update}, \text{Node-Update}\}$.

Example 15. An Update PUDG can be used to model uncertainty that is focused on particular portions of the data-graph. For example, Σ_n, Σ_e can contain distinguished symbols (such as NAME? and ?) which denote unknown data. Our prior $\mathcal{I}$ might represent a distribution of probabilities over the real state of the world, with no data-graph in $\mathcal{I}$ containing one of the uncertainty labels. Then, we could use a noisy observer $\mathcal{R}$ that only modifies a graph by transforming precise data values or labels into unknown ones. See Figure 5 for a toy representation of this kind of modelling.

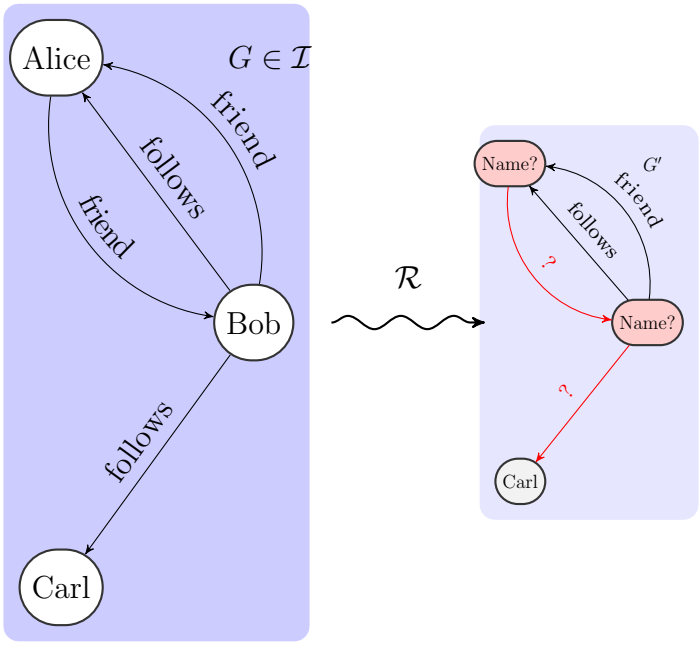
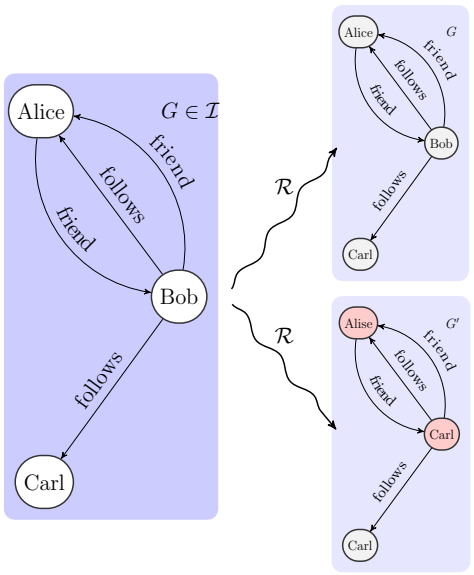
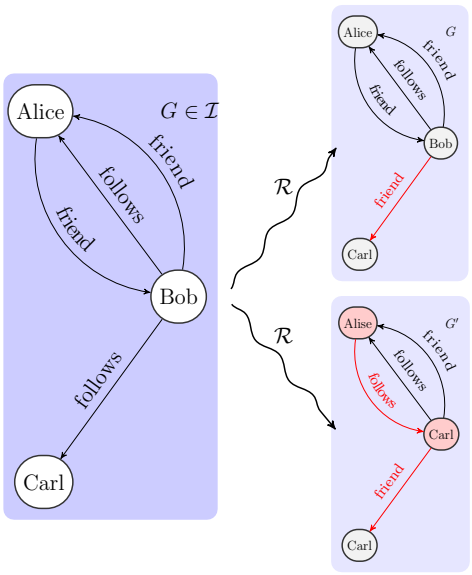


Figure 5: An Update PUGD with distinguished symbols to pinpoint which data is uncertain.



(a) Here the $\mathcal{R}$ is a Node-Update noisy observer, which can introduce changes to the original data values in the nodes, but does not otherwise modify the original data-graph.



(b) Here the $\mathcal{R}$ is a general Update noisy observer. It can introduce changes to both the data in the nodes and to the edge labels, but it does not otherwise modify the original data-graph.

Figure 6: Two types of Update EMDGs, with different restrictions on the function $\mathcal{R}$.

We say that $\mathcal{I}$ is **efficiently computable** if there exists an algorithm $A_{\mathcal{I}}$ and a constant $k_{\mathcal{I}} \in \mathbb{N}$ such that, given any data-graph $G = (V, L, D)$, $A_{\mathcal{I}}$ computes $\mathcal{I}(G)$ in time $\mathcal{O}(|V|^{k_{\mathcal{I}}})$. Analogously, we say that $\mathcal{R}$ is **efficiently computable** if there exists an algorithm $A_{\mathcal{R}}$ and $k_{\mathcal{R}} \in \mathbb{N}$ such that, given any pair of data-graphs $G = (V_G, L_G, D_G)$ and $G' = (V_{G'}, L_{G'}, D_{G'})$, $A_{\mathcal{R}}$ computes $\mathcal{R}(G)(G')$ in time $\mathcal{O}((|V_G| + |V_{G'}|)^{k_{\mathcal{R}}})$. Whenever we are referring to the time complexity of an algorithm that computes efficiently $\mathcal{I}$ and $\mathcal{R}$ as subroutines, we will assume that the time complexity of each call is bounded by $\mathcal{O}(|V|^{k_{\mathcal{I}}})$ and $\mathcal{O}((|V_G| + |V_{G'}|)^{k_{\mathcal{R}}})$, respectively.

Integrity constraints. It is common to expect databases to follow some semantic structure related to the world they intend to represent. This structure can be enforced by defining a set $\mathcal{C}$ of integrity constraints that limit the shape and data contained in the database. In the case of data-graphs, a common way of defining these constraints is through *path constraints* defined in some specific logical language. Given a constraint φ, we want our database to be consistent with respect to φ, given some definition of consistency based on the semantics of φ.

Taking into account that the data model we consider is graph-based, as opposed to the relational case, it is possible to use navigational logics to express specific structures that the data-graphs need to follow. Furthermore, in the context of PUDGs, it is possible to use integrity constraints to define the underlying probabilistic data-graph $\mathcal{I}$: based on prior knowledge, a user might know that the clean database satisfies some topological structure captured by an expression α, and therefore define

$$\mathcal{I}_\alpha(G) = \begin{cases} \frac{1}{N(G,\alpha)} & G \models \alpha \\ 0 & \text{otherwise} \end{cases}$$

where $N(G, \alpha)$ is a normalization factor required to satisfy that $\sum_{G \in U} \mathcal{I}(G) = 1$. Then, the possible clean states are precisely those that satisfy the topological restriction α.

In this work, constraints will be defined with the **Reg-GXPath** language. These formulas can capture both nodes and pairs of nodes based upon expressions that allow to define paths along the graph with common regular expression operators as well as being able to compare data values.

Path expressions of Reg-GXPath are given by:

$$\alpha, \beta = \epsilon \mid \_ \mid \mathrm{A} \mid \mathrm{A}^- \mid [\varphi] \mid \alpha \cdot \beta \mid \alpha \cup \beta \mid \alpha \cap \beta \mid \alpha^* \mid \overline{\alpha} \mid \alpha^{n,m}$$

where A iterates over all labels from Σ_e and φ is a *node expression* defined by the following grammar:

$$\varphi, \psi = \neg\varphi \mid \varphi \wedge \psi \mid \langle\alpha\rangle \mid \mathrm{c}^{=} \mid \mathrm{c}^{\neq} \mid \langle\alpha = \beta\rangle \mid \langle\alpha \neq \beta\rangle \mid \varphi \vee \psi$$

where α and β are path expressions (and so they are defined by mutual recursion), and c iterates over Σ_n. If we only allow the Kleene star to be applied to labels and their inverses (i.e., A^-) we then have a subset of Reg-GXPath called Core-GXPath. The semantics for these languages are defined in [37] in a similar fashion as the usual regular languages for navigating graphs, [10], while adding some extra capabilities such as the complement of a path expression $\overline{\alpha}$ and data tests. The $\langle\alpha\rangle$ operator is the usual one for *nested regular expressions* or NREs used in [13]. Given a data-graph $G = (V, L, D)$, the semantics is defined as follows:

$$\begin{aligned}
&[\![\epsilon]\!]_G = \{(v, v) \mid v \in V\} \\
&[\![_]\!]_G = \{(v, w) \mid v, w \in V, L(v, w) \neq \emptyset\} \\
&[\![\mathrm{A}]\!]_G = \{(v, w) \mid \mathrm{A} \in L(v, w)\} \\
&[\![\mathrm{A}^-]\!]_G = \{(w, v) \mid \mathrm{A} \in L(v, w)\} \\
&[\![\alpha^*]\!]_G = \text{the reflexive transitive closure of } [\![\alpha]\!]_G \\
&[\![\alpha.\beta]\!]_G = [\![\alpha]\!]_G \circ [\![\beta]\!]_G \\
&[\![\alpha \cup \beta]\!]_G = [\![\alpha]\!]_G \cup [\![\beta]\!]_G
\end{aligned}$$

$$[\![\alpha \cap \beta]\!]_G = [\![\alpha]\!]_G \cap [\![\beta]\!]_G$$
$$[\![\overline{\alpha}]\!]_G = V \times V \setminus [\![\alpha]\!]_G$$
$$[\![[\varphi]]\!]_G = \{(v,v) \mid v \in [\![\varphi]\!]_G\}$$
$$[\![\alpha^{n,m}]\!]_G = \bigcup_{k=n}^{m} ([\![\alpha]\!]_G)^k$$
$$[\![\langle\alpha\rangle]\!]_G = \pi_1([\![\alpha]\!]_G) = \{v \mid \exists w \in V\, (v,w) \in [\![\alpha]\!]_G\}$$
$$[\![\neg\varphi]\!]_G = V \setminus [\![\varphi]\!]_G$$
$$[\![\varphi \wedge \psi]\!]_G = [\![\varphi]\!]_G \cap [\![\psi]\!]_G$$
$$[\![\varphi \vee \psi]\!]_G = [\![\varphi]\!]_G \cup [\![\psi]\!]_G$$
$$[\![\mathrm{c}^{=}]\!]_G = \{v \in V \mid D(v) = \mathrm{c}\}$$
$$[\![\mathrm{c}^{\neq}]\!]_G = \{v \in V \mid D(v) \neq \mathrm{c}\}$$
$$[\![\langle\alpha = \beta\rangle]\!]_G = \{v \in V \mid \exists v', v'' \in V,\, (v,v') \in [\![\alpha]\!]_G,\, (v,v'') \in [\![\beta]\!]_G,\, D(v') = D(v'')\}$$
$$[\![\langle\alpha \neq \beta\rangle]\!]_G = \{v \in V \mid \exists v', v'' \in V,\, (v,v') \in [\![\alpha]\!]_G,\, (v,v'') \in [\![\beta]\!]_G,\, D(v') \neq D(v'')\}$$

Example 16. For example, if G is the data-graph of Figure 1, then $[\![\textsc{Dave}^{=}]\!]_G$ would be the singleton set $\{2\}$, containing the (only) node with data value DAVE.

We also have that $[\![\textsc{friend}[\textsc{Dave}]]\!]_G = \{(1,2),(3,2)\}$, representing the pairs of nodes such that we can arrive from the first node to the second one via a path of the form FRIEND[DAVE], which in this case indicates that we move through a FRIEND edge to end up in a node with data DAVE.

And if we want to express 'persons that are following at least two persons with different names', this can be captured by the node expression: $\langle\textsc{follows} \neq \textsc{follows}\rangle$, whose semantics can be read in a more generic manner as 'From this node, it is possible to descend via the FOLLOW edge on one hand, via the FOLLOW edge on the other, and reach two nodes with different data values'. In our running example data-graph, the only node that satisfies this expression is node 4.

We use $\boldsymbol{\alpha \Rrightarrow \beta}$ to denote the path expression $\beta \cup \overline{\alpha}$, and $\boldsymbol{\varphi \Rightarrow \psi}$ to denote the node expression $\psi \vee \neg\varphi$. We also note a label $\mathbf{a}$ as $\downarrow_{\mathrm{A}}$ in order to easily distinguish the 'path' fragment of the expressions. For example, the expression $\textsc{son\_of}[\textsc{Maria}^{=}]\textsc{sister\_of}$ will be noted as $\downarrow_{\textsc{son\_of}} [\textsc{Maria}^{=}] \downarrow_{\textsc{sister\_of}}$.

Naturally, the expression $\alpha \cap \beta$ can be rewritten as $\overline{\overline{\alpha} \cup \overline{\beta}}$ while preserving the semantics, and the same holds for operators $\wedge$ and $\vee$ for the case of node expressions using the $\neg$ operator. In this grammar, all these operators are defined given that there are some natural restrictions of the language which have the same grammar except for some "negative" productions (like $\overline{\alpha}$), in which these simulations would not be possible, and therefore we need to consider them separately in the definition. Considering only the "positive" fragment of a language usually allows obtaining better complexity bounds in reasoning problems. This happens mainly because of the *monotonicity* property: intuitively, a set of expressions from a language $\mathcal{L}$ over structures $\mathcal{H}$ is said to satisfy monotony if whenever $H \in \mathcal{H}$ satisfies $\nu \in \mathcal{L}$, then H' satisfies ν for every $H \subseteq H'$.

In our case, **Reg-GXPath^{pos}** is defined as the subset of Reg-GXPath expressions that do not use $\overline{\alpha}$ nor $\neg\varphi$. Thus, in Reg-GXPath^{pos} we will not be able to simulate the $\cap$ operator unless it is present in the original Reg-GXPath grammar. Moreover, it can be shown that the set of Reg-GXPath^{pos} node expressions satisfy monotony in the following sense: if $v \in [\![\nu]\!]_G$ for $\nu \in$ Reg-GXPath^{pos}, then $v \in [\![\nu]\!]_{G'}$ for every data-graph $G' \supseteq G$.

Given a Reg-GXPath formula η, we denote by $\boldsymbol{\Sigma_n^{\eta}}$ the set of data values that are mentioned in η. More precisely, $\Sigma_n^{\eta} = \{\mathrm{c} \in \Sigma_n : \mathrm{c}^{=}$ or $\mathrm{c}^{\neq}$ is a subexpression of $\eta\}$. Note that $|\Sigma_n^{\eta}| \leq |\eta|$.

4 Data cleaning

One of the intuitive ideas behind an EMDG $(\mathcal{I}, \mathcal{R})$ is that $\mathcal{I}$ represents a known distribution of 'possible worlds', while $\mathcal{R}$ represents how each of them could be 'observed' in a noisy manner with the potential

introduction of errors. A PUDG $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$ adds a concrete observation (namely G') to our model, allowing us to actually reason about the 'underlying reality' given our knowledge of possible worlds and the limits or deficiencies of our observation method. A central question that arises naturally in this interpretation, already mentioned in Example 10, is the functional problem of `Data cleaning`, which aims to find the most probable clean world (i.e., a data-graph in $\mathcal{I}$) given the epistemic model of reality and the observation.

Definition 17 (`Data cleaning`)**.** We define:

PROBLEM: Most likely data-graph INPUT: A Π PUDG $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$ OUTPUT: A data-graph $I \in \mathcal{I}$ such that the probability of $\mathcal{B}_{\mathcal{I},\mathcal{R}}(G')(I)$ is maximum.

Recall that this data-graph with maximal probability exists since, by definition of $\mathcal{R}$, for a fixed G', $\mathcal{B}_{\mathcal{I},\mathcal{R}}(G')$ is a probabilistic data-graph, and therefore there is a finite number of graphs with probability greater than 0. Note however that, if we consider unrestricted $\mathcal{I}$ or $\mathcal{R}$, then the problem might not be computable.

We also define a decision problem related to data cleaning, DATA CLEANING LOWER BOUND:

PROBLEM: Is there a sufficiently likely clean data-graph? INPUT: A Π PUDG $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$ and a (rational) bound $b \in [0, 1]$ OUTPUT: Decide whether there exists a data-graph $I \in \mathcal{I}$ such that $\mathcal{B}_{\mathcal{I},\mathcal{R}}(G')(I) > b$.

Note that an algorithm for solving the data cleaning problem can be used to solve the decision version by checking whether the data-graph I found by the algorithm satisfies $\mathcal{B}_{\mathcal{I},\mathcal{R}}(G')(I) > b$.

In what follows we study this problem considering Subset, Superset and Update PUDGs, respectively. For each of them we show that the problem is intractable even when considering models that allow to efficiently compute $\mathcal{B}$. Nonetheless, we also find some tractable restrictions that bound the number of data-graphs which derive into the unclean one.

4.1 Data cleaning in Subset PUDG

We start studying the data cleaning problem for Subset PUDGs. Suppose that we have a Subset PUDG $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$, and we want to find the data-graph $I \in \mathcal{I}$ that maximizes $\mathcal{B}(G')(I)$. Note that it follows from Definition 5 that the cosupport of G' is finite, however this condition does not prevent that its cardinal might be of arbitrary size, and thus generating it could result too expensive. Moreover, even if we assume that $\mathcal{I}$ and $\mathcal{R}$ can be computed in polynomial-time on the input size, this is not enough to deduce that $\mathcal{B}$ is also efficiently computable. We prove that even in the case where the cosupports and $\mathcal{B}$ are efficiently computable, the problem can still be intractable as the next theorem shows.

Theorem 18. *There is a fixed EMDG $(\mathcal{I}, \mathcal{R})$ with $\mathcal{I}$, $\mathcal{R}$ and $\mathcal{B}_{\mathcal{I},\mathcal{R}}$ efficiently computable, for which the problem* DATA CLEANING LOWER BOUND *(with G' and b as the only inputs) is* NP-COMPLETE*, even when considering Subset PUGDs with no node deletions (i.e., $\mathcal{R}(G)(G') > 0$ implies $V_G = V_{G'}$).*

Proof. The upper bound can be easily proven by noticing that, since we are not allowing node deletions, if there is an I such that $\mathcal{B}(G')(I) > b$ then $|I| = poly(|G'|)$ and $\mathcal{B}(G')(I)$ can be computed in $poly(|G'| + |I|)$. Now, for the hardness part, we reduce 3-SAT to Subset PUGD with a fixed $\mathcal{I}$ and $\mathcal{R}$.

Intuitively, $\mathcal{I}$ will be a distribution over all data-graphs $G_{\phi,v}$ representing a Boolean formula ϕ alongside an assignment v over the variables of ϕ. Probability $\mathcal{I}(G_{\phi,v})$ will be defined so that it is higher when $v \models \phi$. On the other hand, $\mathcal{R}$ will be defined in such a way that it will simply delete the part of the graph that represents the assignment. Then, given a data-graph G'_ϕ representing a formula without an assignment, the data cleaning problem will consist in finding, if there exists, the assignment v that makes ϕ evaluate to true.

Formally, let $\Sigma_e = \{\text{IS\_LITERAL}, \text{IS\_LITERAL\_NEGATED}, \text{VALUE}, e_1, e_2\}$ and $\Sigma_n = \{\text{VAR}, \text{CLAUSE}, \top, \bot\}$. For every CNF formula ϕ of n variables $x_1, \ldots, x_n$ and m clauses $c_1, \ldots, c_m$ and every assignment $v : var(\phi) \to \{\bot, \top\}$, the representation of ϕ and v is the data-graph $G_{\phi,v} = (V, L, D)$, given by:

1. $V = \{1, 2, \ldots, n+m+2\}$. Elements from 1 to n represent the variables, elements from $n+1$ to $n+m$ represent the clauses, and $n+m+1, n+m+2$ are two special nodes. From now on, we refer to the ith VAR node as the variable x_i, and to the jth CLAUSE node as the c_j clause.
2. $D(x_i) = \text{VAR}$ for every $i = 1, \ldots, n$, $D(c_j) = \text{CLAUSE}$ for every $j = 1, \ldots, m$, $D(n+m+1) = \bot$ and $D(n+m+2) = \top$.
3. For each variable x_i and clause c_j, $L(x_i, c_j) = \{\text{IS\_LITERAL}\}$ if x_i is a literal of c_j, $L(x_i, c_j) = \{\text{IS\_LITERAL\_NEGATED}\}$ if $\neg x_i$ is a literal of c_j.
4. For each variable x_i, $L(x_i, n+m+1) = \{\text{VALUE}\}$ if $v(x_i) = \bot$, or $L(x_i, n+m+2) = \{\text{VALUE}\}$ if $v(x_i) = \top$.
5. $L(n+m+1, n+m+2) = \{e_i\}$ for some $i = 1, 2$.
6. The data-graph has no other edges.

Notice that the assignment v is only relevant in rule (4). If we dropped this rule and (5) we obtain for each ϕ a unique data-graph representation denoted by G'_ϕ.

Given an arbitrary data-graph G it is possible to decide in polynomial time if G has the form $G_{\phi,v}$ and obtain in polynomial time a formula ϕ and an assignment v that it represents. Considering this, we define a distribution across the set of data-graphs in the following way:

$$\mathcal{I}(G) = \begin{cases} \frac{1+\phi_v}{N(\phi)} & \text{if } G = G_{\phi,v} \text{ for some } \phi, v \text{ and } (\bot, e_1, \top) \in E_G \\ \frac{1-\phi_v}{N(\phi)} & \text{if } G = G_{\phi,v} \text{ for some } \phi, v \text{ and } (\bot, e_2, \top) \in E_G \\ 0 & \text{otherwise,} \end{cases}$$

where $N(\phi)$ is a normalization factor based on the number of variables, clauses and assignments of ϕ, that allows the probability function to satisfy $\sum_{I \in U} \mathcal{I}(I) = 1$ and ϕ_v is either 1 or 0, depending on whether $v \models \phi$ or not, respectively. We now show a way to build this normalization factor so that it can be easily computed given a formula ϕ.

First, we define $N(\phi)$ such that, given some natural numbers n and m, the total probability assigned to the set of data-graphs representing formulas with n variables and m clauses is precisely the entry in column n and row m in Table 1. We denote from now on this entry of the table by $T(n, m)$. Note that, if the probabilities are assigned in this way, then $\mathcal{I}$ is a well-defined probabilistic database. Also, any entry $T(n, m)$ can be computed in polynomial-time on n and m.[1]

Now, we will distribute the probability evenly across all data-graphs that represent a formula of n variables and m clauses, and thus the final probability is obtained by multiplying this number by the corresponding value in the table. The final probability assigned is therefore $T(n, m) \times \frac{1}{C(n,m)}$, where[2] $C(n, m) = 2 \times 2^n (8\binom{n}{3})^m$.

[1] It can be shown that the entry in row m and column n has value $2^{-(1+\frac{n(n+1)}{2}+m'(n'+1)+\frac{m'(m'+1)}{2})}$

[2] The formula can be derived in the following way: we need to account for the 2^n possible assignations of the n variables, all the $8\binom{n}{3}$ ways in which each clause can independently pick the literals that appear in it, and finally the 2 distinct labels that the edge between $\bot$ and $\top$ can have.

m \ n	0	1	2	3	$\cdots$
0	$\frac{1}{2^1}$	$\frac{1}{2^2}$	$\frac{1}{2^4}$	$\frac{1}{2^7}$	
1	$\frac{1}{2^3}$	$\frac{1}{2^5}$	$\frac{1}{2^8}$		
2	$\frac{1}{2^6}$	$\frac{1}{2^9}$		$\ddots$	
$\vdots$					

Table 1: Table showing how much of the "probability weight" is assigned into each set of data-graphs, for each value of n and m.

Regarding $\mathcal{R}$, it maps each data-graph G to a uniform distribution across every data-graph H with the same nodes as G but just a subset of its edges. That is, $\mathcal{R}(G)(H) = \frac{1}{2^{|E_G|}}$ if $V_H = V_G$ and $H \subseteq G$, and $\mathcal{R}(G)(H) = 0$ otherwise.

Finally, the reduction is as follows: given a 3-CNF formula ϕ of n (ordered) variables and m (ordered) clauses, we construct the data-graph $G' = G'_\phi$ and define $b = \frac{1}{2^{|E_{G'}|}} \times T(n,m) \times \frac{1}{C(n,m)}$. Clearly, the data-graph can be built in $poly(|\phi|)$ time, and each element in the product that composes b can also be built in time $poly(n+m)$. It is clear that if there exists an assignment v to the variables of ϕ that evaluates it to true, then $\mathcal{B}(G'_\phi)(G_{\phi,v}) > b$, otherwise no data-graph in the cosupport of G'_ϕ will have a probability that is bigger than b. □

This theorem shows that even if we fix both functions $\mathcal{I}$ and $\mathcal{R}$ the problem remains intractable. Note that fixing this part of the input seems reasonable: both functions $\mathcal{I}$ and $\mathcal{R}$ capture domain knowledge and the semantics of the data the data-graph represents.

There are, however, at least two ways in which we can simplify this problem to obtain a polynomial-time resolution.

First, we could try and control the complexity that arises from the relationship between G and the EMDG. It might be the case that we find an interesting restriction of the problem by considering some EMDG that captures topological aspects of the data and a particular subclass of data-graphs. For example, in [11] some database repairing problems are conceptually similar to the ones discussed here, and can be solved in polynomial time when considering graphs with bounded treewidth.

Second, we could control the complexity encapsuled in both $\mathcal{I}$ and $\mathcal{R}$ or by restricting the overall possibilities of those functions as input of the problem.

For instance, if we condition the problem to obtain a polynomial bound on the cardinality of the data-graphs that are able to "transitition" to the observed state G through $\mathcal{R}$, then it becomes tractable under the assumption that $\mathcal{B}$ is efficiently computable. More precisely, we ask for the size of the cosupport of a data-graph G', namely $\sigma(G')$, to be bounded by some polynomial function. We can do this by forbidding node deletions and by bounding the number of edges that can be added by a constant k_e. Formally, we assume that $\mathcal{R}(I)(G') = 0$ if $|E_I \setminus E'_G| > k_e$ or $V_I \neq V_{G'}$. Assuming these restrictions, we can show that $\sigma(G')$ is bounded by a polynomial on $n = |V_{G'}|$:

$$|\sigma(G')| \leq \sum_{\substack{i=0 \\ i \leq \text{missing}(G')}}^{k_e} \binom{\text{missing}(G')}{i} \leq k_e(\text{missing}(G'))^{k_e} = \mathcal{O}(n^{2k_e}), \tag{2}$$

This equation counts all the graphs obtained by adding at most k_e edges to G'. This quantity is represented by each combinatorial number, for every $i \leq \text{missing}(G')$, and $\text{missing}(G') = n^2|\Sigma_e| - E_{G'}$ denotes the number of edges that can be added to G'. The next theorem follows immediately from the previous discussion.

Theorem 19. `Data cleaning` *of Subset PUDG can be computed in polynomial time in the size of the observed data-graph G' whenever the following conditions hold:* (i) *node deletions are not allowed, and* (ii) *the number of edges deleted is bounded by a constant k_e.*

Proof. The algorithm consists of two stages: (a) generating the set C of candidate data-graphs, whose cardinality is bounded by $\mathcal{O}(n^{2k_e})$ due to Equation (2) and (b) evaluating $\mathcal{I}(G) \times \mathcal{R}(G)(G')$ for each $G \in C$. This algorithm's complexity can be bounded by $\mathcal{O}(n^{2k_e} \times (n^{k_\mathcal{I}} + n^{k_\mathcal{R}}))$. □

Though the previous result imposes a considerable restriction to the problem, i.e., bounding the number of possible deletions, the resulting family of realization models is still quite expressive, as they can assign probabilities based on topological properties and on the different present data values and edge labels.

4.2 Data cleaning in Superset PUDG

Now we consider the Superset PUDG version of the Data Cleaning problem. This is, given a Superset PUDG $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$, we want to find the data-graph $I \in \mathcal{I}$ that maximizes $\mathcal{B}_{\mathcal{I},\mathcal{R}}(G')(I)$. Recall that G' is the observed unclean data-graph that may have loops and as many edges as distinct possible labels. Notice that, in this case, and disregarding whether we consider node and/or edge deletions, every graph in I and in the cosupport of G' is necessarily a subgraph of G'. Therefore, the set of data-graphs in $\mathcal{B}(G')$ is finite, and its size is bounded by $\sum_{i=0}^{n} \binom{n}{i} 2^{i^2 \times |\Sigma_e|} \leq 2^{n^2 \times |\Sigma_e| + n}$ ($n = |V_{G'}|$) in the worst case scenario. Moreover, if we only consider edge deletions, then $\sigma(G')$ is bounded by $2^{n^2 \times |\Sigma_e|}$.

As in the case of Subset PUDGs, the fact that $\mathcal{B}_{\mathcal{I},\mathcal{R}}$ is efficiently computable is not enough to conclude that we can find in polynomial time the most probable clean data-graph:

Theorem 20. *There exists a fixed $\mathcal{I}$ and $\mathcal{R}$ such that $\mathcal{B}_{\mathcal{I},\mathcal{R}}$ is efficiently computable and the problem* DATA CLEANING LOWER BOUND *(with only G' and b as the only inputs) is* NP-COMPLETE*, even when $\mathcal{R}$ does not allow node additions*

Proof. The proof is similar to that for Theorem 18. See the appendix for the details. □

As before, this result tells us that even though the search space is now clearly bounded we will need to impose some further restrictions in the structure of both $\mathcal{R}$ and $\mathcal{I}$. the next theorem shows that if we bound the number of nodes and edges that can be added through $\mathcal{R}$ by some constant then the problem is tractable, assuming an efficiently computable $\mathcal{B}$.

Theorem 21. *The Data Cleaning Superset PUDG problem can be solved in polynomial time if $\mathcal{R}(G)(G') > 0$ implies $|V_{G'} \setminus V_G| + |E_{G'} \setminus E_G| < c$ for some fixed $c \in \mathbb{N}$ and any pair of data-graphs G, G'.*

Proof. This can be proven in the same way as Theorem 19. There is only a polynomial number of data-graphs that can be mapped to a positive probability through $\mathcal{R}$ given the observed data-graph G', and they can be easily enumerated. More precisely, the set of candidate data-graphs C is bounded by:

$$|C| \leq \sum_{i=0}^{c} \binom{n}{c} \times \sum_{\substack{i=0 \\ i \leq n^2|\Sigma_e| - \text{missing}(G')}}^{c} \binom{n^2|\Sigma_e| - \text{missing}(G')}{i} = \mathcal{O}(n^{3c})$$

Then, enumerating every data-graph $G \in C$ and evaluating $\mathcal{I}(G) \times \mathcal{R}(G)(G')$ can be done in time $\mathcal{O}(n^{3c} \times (n^{k_\mathcal{I}} + n^{k_\mathcal{R}}))$. □

Observe that this theorem shows that in the superset case we can allow $\mathcal{R}$ to add nodes and still have a polynomial algorithm for solving the data cleaning problem. In the analogous situation of the subset case, we could not allow to remove nodes because when trying to explore the cosupport through $\mathcal{R}$ we would need to guess the data values of the nodes that were deleted, and the possibilities for these are not bounded.

4.3 Data Cleaning in Update PUDGs

As in the previous cases, if we consider efficiently computable $\mathcal{B}$'s, then the problem can still be intractable:

Theorem 22. *There exists a fixed $\mathcal{I}$ and $\mathcal{R}$ such that $\mathcal{B}_{\mathcal{I},\mathcal{R}}$ is efficiently computable and the problem* DATA CLEANING LOWER BOUND *(with only G' and b as the only inputs) is* NP-COMPLETE*, even when $\mathcal{R}$ only allows updating edge labels.*

Proof. Again, the proof on this theorem is similar to the one used for Theorem 20. See the appendix for the details. □

As before, we can bound the size of the cosupport of G' in order to deduce a restriction of the problem that can be solved in polynomial time. We can do this in a natural way by bounding the number of nodes whose data values could be changed by $\mathcal{R}$, in a similar way as in Theorem 21. Moreover, for every data value c, we need a bound on the number of data values that can be updated to c through $\mathcal{R}$. We can formalize this by requiring the existence of a constant k and a **k-data-prior** function $f : \Sigma_n \to \mathcal{P}(\Sigma_n)$ computable in polynomial time on the representation of the input data value such that $|f(c)| \leq k$, for every $c \in \Sigma_n$. Then, given a function f with the previous restrictions, we can consider Update EMDGs $(\mathcal{I}, \mathcal{R})$ such that $\mathcal{R}$ agrees with f in the following way: if $\mathcal{R}(G)(G') > 0$, then for every $v \in V_{G'}$ holds that $D_G(v) \in f(D_{G'}(v))$. Hence, we can prove the following result:

Theorem 23. *Given a k-data-prior function f as input such that $\mathcal{R}$ depends on f, if $\mathcal{R}$ is allowed to update only a constant number z of data values, then the Data Cleaning Node Update PUDG problem can be solved in polynomial time.*

Proof. Given the observed unclean data-graph G', we generate the cosupport of G' given $\mathcal{R}$ using the k-data-prior function f. In particular, the size of the cosupport is bounded by $\binom{n}{z}k^z = \mathcal{O}(n^z)$, and it can be generated in polynomial time on its size. Then, we only need to evaluate $\mathcal{B}$ and keep the one with the highest probability. All of this can be done in $O(n^z \times (n^{k_\mathcal{I}} + n^{k_\mathcal{R}}))$. □

In the following section we propose a series of restrictions over the elements of the framework in order to obtain lower complexity results.

5 Global Cardinality Constraints

We now focus on other restrictions of the Data Cleaning problem that are particularly interesting within the framework of data-graphs, as they allow modifying the data values on the nodes. In particular, in this section we consider Node-Update PUDGs.

Suppose that we have a prior on how many times each edge label e or data value c is present in the original data-graph. In this scenario, we consider as a preferred answer those data-graphs that are closer to these priors. In other words, we consider cardinality constraints surrounding the number of appearances of one or more predetermined data subsets of values and/or edge labels.

First, we present some tractable versions based on cardinality constraints. Afterward, we explore other restrictions to the problem that make it remain hard. This allows us to shed light on the line between tractability and hardness for Node-Update EMDG.

5.1 Data cleaning for data-graphs with subsets of fixed data

Let us consider the following restriction of the general Node-Update EMDG model, where data in nodes are unique and belong to a predefined subset of "valid" data values. Let $G' = (V, L, D)$ be our observed unclean data-graph, suppose that $|\Sigma_n| = |V| = N$ and $D(V) = \Sigma_n$, and that Σ_n is part of the input. Let $m < N$, and let $f_{G'} : [1, m] \to V$ be an injective function that indicates for which nodes of the observed graph we know with certainty that their data-values are correct. More precisely, $\mathcal{B}(G')(I) = 0$ if I modifies

a node in the image of f. In other words, we have certainty over the data values for precisely m nodes of G', whereas for the remainder of the nodes we do not. Suppose we are given a weight function such that these weights determine the probability of a certain assignment of data-values to each node. More precisely, let $w : \Sigma_n \times V \to [0,1]$ such that $\sum_{I \in \mathcal{I}} \prod_{i,j=1}^{N} w_{ij} p_{ij}^{(I)} = 1$, where

$$p_{ij}^{(I)} = \begin{cases} 1 & \text{if the data-value } i \text{ has been assigned to the node } j \\ 0 & \text{if not} \end{cases}$$

and $\sum_{i=1}^{N} p_{ij}^{(I)} = 1$ for every $j = 1, \ldots, N$ for every $I \in \mathcal{I}$. Moreover, suppose that $\mathcal{R}(H)(G') = 1$ for every $H \in \mathcal{I}$. Considering all these assumptions, the value $\mathcal{B}(G')(I)$ is:

$$\mathcal{B}(G')(I) = \frac{\mathcal{I}(I) \times \mathcal{R}(I)(G')}{\sum \mathcal{I}(H) \times \mathcal{R}(H)(G')} = \frac{\mathcal{I}(I)}{\sum \mathcal{I}(H)} = \mathcal{I}(I) = \prod_{(i,j) \in \Sigma_n \times (V \setminus Im(f))} w_{ij} p_{ij}^{(I)}.$$

Since we are looking for the graph with the maximum probability, this is equivalent to finding those weights for which the product is maximum. This allows us to take into consideration the adjacencies and labels in our observed unclean data-graph when assigning the weights. Additionally, we could also ask for an extra feature assuming that $\sum_{j=1}^{N} w_{ij} = 1$ for every $j = 1, \ldots, N$. This condition can be interpreted as: "all the possible data value assignments for each node determine a probability by itself". In summary, we define the problem `Data cleaning (Fixed data)` as follows:

PROBLEM: `Data cleaning (Fixed data)` INPUT: A set of data-values Σ_n, a PUDG $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$ over the set of data-values Σ_n, an injective function $f_{G'} : [1,m] \to V_{G'}$, and a weight function $w : \Sigma_n \times V(G') \to [0,1]$ as described above. OUTPUT: A data-graph $I_m \in \mathcal{I}$ such that the probability of $\mathcal{B}(G')(I_m)$ is maximum.

Theorem 24. *The problem* `Data cleaning (Fixed data)` *can be solved in* $\mathcal{O}(n^3)$ *time.*

Proof. To determine the complexity of this problem, we can reinterpret it as an N-rooks problem as follows. Consider an N-square board, where the ranks (i.e. the rows) represent the data values and the files (i.e. the columns) represent the nodes of G'. We place a rook in square (i,j) if we are certain that the node j has the data value i. Thus, there are m rooks with a fixed position in our board, all of them in non-attacking positions since we assumed that these m values are distinct and assigned to different nodes. Now, consider that our board has assigned weights w_{ij} to each square, which represents the probability of the assignment of each data value to each node. We can determine the remaining positions of the $N - m$ rooks by considering an auxiliary graph H as follows. Let H be a graph that has its vertices partitioned into V_1 and V_2, having N vertices each: one of them –let us say V_1– has one vertex for each data value, and the other one has one vertex for each node. There are no edges between vertices in the same partition, and there is an edge between x_i in V_1 and y_j in V_2 if and only if the data value i can be assigned to the node j with positive probability. More precisely, we consider that the edge $x_i y_j$ has a weight w_{ij}, this is the weights of the edges between vertices in V_1 and V_2 are now the probabilities determined on each square. Hence, solving Node-Update in this case derives from finding a maximal weighted matching in the bipartite graph H', which can be done in $\mathcal{O}(n^3)$ time [33]. □

5.2 Weak cardinality constraint over data values

We could consider an alternative scenario in which the function $\mathcal{I}$ codifies a cardinality preference criteria over the data values by having a function $T : U \times \Sigma_n \to \mathbb{N}_0$ such that $\sum_{\mathrm{C} \in \Sigma_n} T(G, \mathrm{C}) = |V_G|$ for all G. This function T tells us the number of data values that we want to have for each data value C. Naturally, $T(G, \mathrm{C})$

will be bigger than 0 for at most $|V_G|$ different data values. Furthermore, we require that, if $G = (V, L, D_G)$ and $H(V, L, D_H)$, then $T(G, \text{C}) = T(H, \text{C})$. This condition implies that the cardinality preference depends on the topology, and not on the data values.

Given a function T and a data-graph H we define the penalization factor of H with respect to T as

$$d_T(H) = \sum_{\text{C} \in \Sigma_n} |T(H, \text{C}) - count(H, \text{C})|$$

where $count(H, \text{C})$ denotes the number of times the data value C is present in H. We say that $\mathcal{I}$ *depends on* T if for every pair of data-graphs H, H' with the same set of nodes and edges it is true that $\mathcal{I}(H) > 0$ if and only if $d_T(H) = 0$ and $\mathcal{I}(H), \mathcal{I}(H') > 0$ implies $\mathcal{I}(H) = \mathcal{I}(H')$. This basically means that $\mathcal{I}$, given a set of nodes V, distributes the probability evenly across all data-graphs with the same set of nodes and edges that also satisfy the constraints defined by the function T.

Notice that this prioritizes data-graphs that follow a cardinality criterion over the set Σ_n, completely ignoring the uncleaned observed data-graph G'. We add a simple local realization model $\mathcal{R}$ between clean and unclean data-graphs by using a function $\delta : \Sigma_n \times \Sigma_n \to \mathbb{N}_0$ that defines, for each pair of data values $(\text{C}, \text{D}) \in \Sigma_n^2$, the 'cost' of changing C to D. Naturally, we impose that $\delta(\text{C}, \text{C}) = 0$ for every $\text{C} \in \Sigma_n$. In other words, $\mathcal{R}$ prioritizes those data-graphs H such that the cost of the data values transitions required to go from H to the unclean data-graph G is minimized.

More formally, given two data-graphs H and G such that $V_H = V_G, L_H = L_G$ and a weight function δ between data values, we define the **cost** of going from H to G as

$$c_\delta(H, G) = \sum_{v \in V_H} \delta(D_H(v), D_G(v)).$$

We say that $\mathcal{R}$ *depends on* δ if $\mathcal{R}(H)(G) > \mathcal{R}(H')(G')$ if and only if $c_\delta(H, G) < c_\delta(H', G')$.

Observe that the following theorem is (almost) a generalization of the scenario described in the previous section. In that case, the weight w of the transition between data values could also depend on the involved vertex.

Theorem 25. *The* `Data cleaning` *problem can be solved in* $O(n^3)$ *if* $\mathcal{I}$ *depends on a given function* T *and* $\mathcal{R}$ *depends on a given local function* δ.[3]

Proof. Given the Update PUDG $(\mathcal{I}, \mathcal{R}, G)$ where $\mathcal{I}$ depends on T and $\mathcal{R}$ depends on δ, we know that the most probable clean data-graph will be the one satisfying the constraints defined by T and that minimizes the cost of the transitions as defined in δ. We define the complete bipartite undirected graph $I = (V_I, E_I)$ in such a way that a minimum weighted perfect matching in I codifies the cleanest data-graph H:

$$V_I = \{v : v \in V_G\} \cup \{u_{\text{C},i} : \text{C} \in \Sigma_n, T(G, \text{C}) > 0, 1 \leq i \leq T(G, \text{C})\}$$
$$E_I = \bigcup_{v \in V_G} \{(v, \delta(D_G(v), \text{C}), u_{\text{C},i}) : u_{\text{C},i} \in V_I\}$$

The graph I has two sets of nodes: the ones from G and new $T(G, \text{C})$ nodes for every $\text{C} \in \Sigma_n$ such that $T(G, \text{C}) > 0$ (it follows from $\sum_{\text{C} \in \Sigma_n} T(G, \text{C}) = N$ that there are N nodes of the form $u_{\text{C},i}$). Observe that a matching in $\mathcal{I}$ is just an assignation of the nodes from G to the desired data values.

It is easy to see that the assignation with the lowest cost corresponds to the matching with minimum weight. Then, it can be computed in $O(n^3)$. □

This model does not interact with the edges of the data-graph but simple interactions with the edges can be imposed by increasing the expressive power of the realization model: for example, we could consider local functions δ that take as input the edges labels outgoing the nodes being changed, instead of only the data values. As far as these interactions are independent of the data values, the modelling through weighted bipartite matching will remain valid.

[3]Here, we assume that T and δ are functions that can be computed in $O(1)$.

5.3 Data cleaning with valid data preselection

For this next restricted version, let us consider the alphabet Σ_n of data values and, for each node v in G', suppose we are given a subset of data values $X_v \subseteq \Sigma_n$. We may consider that these subsets comprise all the valid data values for each node, trusting in some pre-processing or prior knowledge on the data. For our problem, we assume that the data value of each v in G' is missing. However, we do know that data-graphs G such that $\mathcal{B}(G')(G) > 0$ are precisely those isomorphic to G' for which the data values of each v lies in X_v, following the intuitive idea that these data values represent the only valid assignments for each node. Furthermore, those data-graphs in the cosupport of G' that have the highest probability are those that have fewer distinct data values.

We define the problem `Data cleaning` with valid data preselection when the input is a PUDG $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$ with an $\mathcal{R}$ as above, and a family $\{X_v\}_{v \in G'}$ such that $X_v \subseteq \Sigma_n$ and $|X_v| > 0$.

Theorem 26. *The problem* `Data cleaning` with valid data preselection *is NP-hard even if* $|X_v| \leq 2$ *for every* $v \in G$.

Proof. It follows from the definition that finding a solution of maximum probability for this version of Node-Update EMDG problem is equivalent to finding a Hitting Set for the family $\{X_v\}_{v \in G'}$, since we are looking for a minimum subset of data values $X \subset \Sigma_n$ such that $X \cap X_v \neq \emptyset$ for every $v \in G'$. Moreover, the decision version of this problem is NP-complete even if $|X_v| \leq 2$ [28]. □

Notice that we are considering a quite restricted version of the general Node-Update EMDG problem, still focusing the restrictions on cardinality-based constraints. And even in the very specific case in which we consider every subset X_v of size smaller than 2, we obtain an NP-complete version of our original problem.

We could also consider the analogous version for Edge-Update EMDG. More precisely, we are given subset of edge-labels $X_{v,w} \subseteq \Sigma_e$ for each edge (v, E, w) in $E(G')$. As explained in the previous paragraphs, we consider that the subset $X_{v,w}$ comprises all those valid edge-label values for each edge (v, E, w), once again trusting in some pre-processing or prior knowledge. Moreover, we assume that all the edge-labels are missing, and that the data-graphs G for which $\mathcal{B}(G')(G) > 0$ are precisely those isomorphic to G' such that the edge-labels E of each edge between v and w lie in $X_{v,w}$. Furthermore, those data-graphs in the preimage of G' that have the highest probability are those that have fewer distinct edge-labels. The hardness of this version follows analogously as in the Node-Update case.

Theorem 27. *The problem* `Data cleaning` with valid edge-label preselection *is NP-hard even if* $|X_{v,w}| \leq 2$ *for every* $(v, \mathrm{E}, w) \in E_G$.

In previous sections, we focused on Data Cleaning only considering the epistemic information contained in the PUDG. In the next section, we explore ways to add domain integrity constraints to our model.

6 PUDG with Reg-GXPath constraints

In this section, we consider alternative ways to add constraints to our model using Reg-GXPath expressions. In contrast to the previous section, we now define a way to modify the prior probabilities from $\mathcal{I}$ without modifying the noisy observer $\mathcal{R}$ from the original EMDG.

6.1 Expression-based Constraints in Node-Update PUDGs

We now explore some particular instances of the Data cleaning problem in Node-Update PUDGs. As already mentioned, we focus on this kind of PUDGs given that data values are one of the main features of our data-graph model.

An alternative way to define a system of priorities with the distribution $\mathcal{I}$ is by using a set of *path expressions* or *integrity constraints* that can be evaluated on a data-graph. For example, we consider an

expression η from a language such as Reg-GXPath (see Section 3 for the basic definitions) and define $\mathcal{I}_\eta$ in some way such that $\mathcal{I}_\eta(G) < \mathcal{I}_\eta(G')$ whenever $G \not\models \eta$ and $G' \models \eta$. Then, we prioritize those data-graphs that satisfy some topological structure that we are expecting to see in the clean database, which is expressible through some language $\mathcal{L}$.

We may also 'evaluate' the expression η only in a specific node (sometimes called the origin o) or in a pair of nodes, as it was first proposed in [2] regarding path constraints. This restriction might turn reasoning problems easier (see for example [11]). From now on, we denote the universe of data-graphs that have a distinguished node from where node expressions can be evaluated as U_o, and given $G \in U_o$ we denote this distinguished node as o_G.

Nonetheless, we point out some observations regarding the relationship between the semantics when evaluating an expression in all nodes and when only considering an origin o.

Observation 28. *Relation between global and origin semantics.*

1. *There exists a function $f : U \times$ Reg-GXPath$^{pos} \to U'_o \times$ Reg-GXPath^{pos} computable in polynomial time such that for any data-graph G and Reg-GXPath^{pos} node expression φ is true that $G \models \varphi$ if and only if $G', o_{G'} \models \varphi'$, where $f(G, \varphi) = (G', \varphi')$.*

2. *There exists a function $g : U'_o \times$ Reg-GXPath$^{pos} \to U \times$ Reg-GXPath^{pos} computable in polynomial time such that for any $G \in U_o$ and Reg-GXPath^{pos} expression φ' is true that $G', o_{G'} \models \varphi'$ if and only if $G \models \varphi$, where $g(G', \varphi') = (G, \varphi)$*

Proof. For (1), we can construct G' in the following way: consider an arbitrary order $p_1, \ldots, p_n$ of the nodes of G and a new node p_0 with any data value. Now, add p_0 to G with a loop on a fresh edge label *loop* and define a Hamiltonian cycle over G by using another new edge label ϵ and adding edges (p_i, ϵ, p_{i+1}) for every $0 \leq i \leq n$ where $p_{n+1} = p_0$. Observe that $G \models \eta$ if and only if $G', p_0 \models \langle (\downarrow_\epsilon [f(\eta)])^* \downarrow_\epsilon \downarrow_{loop} \rangle$, where $f(\varphi)$ denotes the expression φ after replacing any $\_$ subformula by $\bigcup_{l:\Sigma_e} \downarrow_l$.[4]

For (2), we build G by adding to every node of G' different from o a loop with edge label *loop*. It follows that $G', o \models \varphi'$ if and only if $G \models \downarrow_{loop} \cup [f(\varphi')]$ where f is the same function as in (1) that replaces the use of the wildcard $\_$.[5] □

Remark 29. When considering path expressions instead of node expressions, there is a result analogous to Observation 28 that relates global semantics with bi-pointed semantics. Indeed, it is easy to see that $G, x, y \models \alpha$ if and only if $G' \models \alpha'$, where G' is constructed by adding fresh edge labels I, F (expanding Σ_e), and adding an edge I from all the nodes to x, and an edge F from y to all the nodes. Then, we consider $\alpha' = \downarrow_\mathrm{I} \alpha \downarrow_\mathrm{F}$.

For the other direction, we have that $G \models \alpha$ if and only if $G', x, y \models \alpha'$, where G is modified into G' by adding two fresh nodes x and y, adding edges with a fresh label I from x towards all the nodes in G, and an edge with fresh label F from all the nodes in G to y. Then, we consider $\alpha' = \overline{\downarrow_\mathrm{I} \overline{\alpha} \downarrow_\mathrm{F}}$.

Notice, nonetheless, that for these results we had to use negation to build the new expression. Then, if we want to remain in the positive fragment, we need to consider only node expressions.

Observation 28 implies that we may interchange the semantics considered for our reasoning problems, as long as the new node expression belongs to the same set of expressions $\mathcal{L}$ and we consider Node-Update PUDGs. Notice that, to go from the global case to the origin one –when considering node expressions– we had to use the Kleene operator over a path expression involving nested sub-expressions and a data test. Any language that is useful for defining constraints in data-graphs must have data-tests, and therefore a reasonable set of expressions to study over the origin semantics whose results do not follow naturally from the global scenario could be the subset of Reg-GXPath^{pos} formulas that only use the Kleene star in restricted

[4] This is necessary to prevent the formula φ from using the edges $\downarrow_e$ in G'.

[5] In both directions, the sets Σ_n and Σ_e might need to be extended in order to find a fresh data value or edge label

cases. In fact, we will show that considering a restricted version of this operator allows for polynomial algorithms to solve the DATA CLEANING problem in the context of Node-Update PUDGs.

Naturally, the difficulty of finding the clean database H that maximizes $\mathcal{B}_{\mathcal{I},\mathcal{R}}(H,G)$ will depend on the complexity of the language $\mathcal{L}$ used to define the expressions. For example, if Reg-GXPath$^{pos} \subseteq \mathcal{L}$ then the problem is intractable:

Theorem 30. *There are fixed finite sets Σ_n and Σ_e, and a fixed Reg-GXPath^{pos} formula η such that given a data-graph G, the problem of deciding if there exists a data-graph H isomorphic to G up to its data values such that $H \models \mu$ is* NP-COMPLETE.

Proof. The problem is in NP: we can guess a data-graph H and check that $H \models \eta$ and $G \equiv H$. Now, we reduce our problem to 3SAT to prove *hardness*.[6]

Given a 3SAT boolean formula ϕ of n variables $x_1 \dots x_n$ and m clauses $c_1 \dots c_m$ we will build a data-graph G and a Reg-GXPath^{pos} node expression φ such that ϕ is satisfiable if and only if there exists a data-graph G' such that $G' \equiv G$ and $G' \models \varphi$.

Consider $\Sigma_n = \{\text{VAR}, \text{CLAUSE}, \bot, \top\}$ and $\Sigma_e = \{\downarrow_-, \downarrow_+, \downarrow_{\text{NOTCLAUSE}}\}$ and define $G = (V, L, D)$ as:

$$
\begin{aligned}
&V = \{x_i : 1 \le i \le n\} \cup \{c_j : 1 \le j \le m\}\\
&L(x_i, c_j) = \{\downarrow_+\} \iff x_i \text{ is a literal from } c_j\\
&L(x_i, c_j) = \{\downarrow_-\} \iff \neg x_i \text{ is a literal from } c_j\\
&L(x_i, x_i) = \{\downarrow_{\text{NOTCLAUSE}}\}\\
&L(z_1, z_2) = \emptyset \text{ for any other case}\\
&D(c_i) = \text{CLAUSE}\\
&D(x_i) = \text{VAR}
\end{aligned}
$$

This data-graph encodes the formula ϕ through the edges $\downarrow_+$ and $\downarrow_-$. Observe that, given a valuation of the variables of ϕ, we can set the data values of the node variables of G accordingly to $\bot$ or $\top$ to represent that valuation. Moreover, we can build a formula that only captures those data-graphs representing an assignment that satisfies ϕ with the node expression

$$\varphi_1 = \langle \downarrow_+^- [\top] \rangle \vee \langle \downarrow_-^- [\bot] \rangle \vee [\text{CLAUSE}^{\neq}].$$

This formula forces every node with data value CLAUSE to have a path through edges $\downarrow_-^-$ and $\downarrow_+^-$ to a value $\bot$ or $\top$, respectively.

We use the NOTCLAUSE edges to forbid any isomorphic data-graph H from changing the data value of the clause nodes in the following way:

$$\varphi_2 = \langle \downarrow_{\text{NOTCLAUSE}} \rangle \vee [\text{CLAUSE}^{=}]$$

That is, we only want to consider data-graphs in which the data values that change are those from the variable nodes, which should remain the same, or change to either $\top$ or $\bot$.

Finally, we set $\varphi = \varphi_1 \wedge \varphi_2$. It follows that ϕ is satisfiable if and only if there exists a data-graph G' such that $G' \equiv G$ and $G \models \varphi$. Observe that given ϕ we can build G in linear time, and therefore we reduced 3SAT to our problem. □

Given a data-graph G, the problem of deciding the existence of an isomorphism G' of G that satisfies a certain formula η may be interpreted as a particular relaxation of the data cleaning problem, in which the only data-graphs with positive weights are those that satisfy η and $\mathcal{R}(G')(G) = 1$ for every data-graph H.

[6] Here we use the assumption that data values are $\mathcal{O}(1)$ to find a polynomial-sized witness. Nevertheless, this theorem remains true in a more general scenario where we allow the size of data values to grow logarithmically because, if there is a repair of G that works as a witness, then there is another that uses the data values in Σ_n^η and at most $|V_G|$ more, where these other data values not in Σ_n^η can be chosen arbitrarily (see Lemma 34).

Moreover, these data-graphs also have the same weight, and therefore are "equally likely" to be the clean data-graph. To understand this "lower bound"' to the data cleaning problem in the context of expression-based constraints we study the following problem:

PROBLEM: ISOMORPHIC-REPAIR INPUT: A data-graph G and a formula η from some set of expressions $\mathcal{L}$ OUTPUT: Decide whether there exists a data-graph H such that $H \equiv G$ and $H \models \eta$.

This problem is similar to the *repair computing* problem, deeply studied in the database theory community (see [15]).

We already showed that the problem is intractable if Reg-GXPath$^{pos} \subseteq \mathcal{L}$. Furthermore, in Theorem 30 we considered the node expression *fixed*. This is a common simplification, usually referred to as the *data complexity* of the problem [51]. Also, notice that in Observation 28 the formula constructed in the first remark has fixed size if η is fixed. Therefore, we conclude that:

Corollary 31. *The problem* ISOMORPHIC REPAIR *is* NP-COMPLETE*, even if the node expression $\eta \in$ Reg-GXPath^{pos} is fixed and the expression is evaluated only in an origin o from V_G.*

As previously remarked, one way to weaken the language $\mathcal{L}$ in order to avoid concluding intractability as a consequence of Theorem 30 and Observation 28, would be to limit the cases in which we allow the Kleene star. If we completely remove it, then the problem remains hard, but only under very particular assumptions:

Theorem 32. *The* ISOMORPHIC-REPAIR *is* NP-COMPLETE*, even if we only evaluate the expression at an origin o, as long as we allow:*

- *The set of data values to be infinite (i.e. $|\Sigma_n| > \infty$)*
- *The expression not to be fixed, and the use of the $\langle\cdot\rangle$ operator and equality data tests ($\text{C}^{=}$).*

Proof. Membership in NP follows from Theorem 30. We prove the hardness by reducing HAMILTONIAN PATH from a distinguished vertex v to our particular case of ISOMORPHIC REPAIR.

Let $\Sigma_n = \mathbb{N}_0$ and $\Sigma_e = \{\text{DOWN}\}$. Given a directed graph $G = (V, E)$ we construct the data-graph $G' = (V', L', D')$ with origin v as follows:

$$V' = V$$
$$L(z, w) = \{\text{DOWN}\} \iff (z, w) \in E \ \forall z, w \in V$$
$$D(z) = 0 \ \forall z \in V.$$

We define the node expression:

$$\varphi = \langle [1^{=}] \downarrow_{\text{DOWN}} [2^{=}] \downarrow_{\text{DOWN}} [3^{=}] \dots \downarrow_{\text{DOWN}} [\text{N}^{=}] \rangle$$

It is easy to see that we can find a new data-graph H equal to G up to its data values such that $H, v \models \varphi$ if and only if G has a Hamiltonian path starting at v. □

The formula used for this proof is extremely simple, however it does depend on the input graph and it strongly relies on Σ_n being infinite.

Now, let us consider a weaker family of expressions, namely the ones defined by the grammar:

$$\alpha, \beta = \epsilon \mid \text{A} \mid \text{A}^{-} \mid [\varphi] \mid \alpha \cdot \beta \mid \alpha \cup \beta \mid \alpha \cap \beta \mid \text{A}^{*} \mid \text{A}^{-*} \mid \alpha^{n,m}$$
$$\varphi, \psi = \varphi \wedge \psi \mid \langle \alpha \rangle \mid \text{C}^{=} \mid \text{C}^{\neq} \mid \langle \alpha = \beta \rangle \mid \langle \alpha \neq \beta \rangle \mid \varphi \vee \psi.$$

This is a subset of Reg-GXPath called Reg-GXPath$^{pos}_{core}$ [37].

Observe that we removed the operation that allowed us to use the Kleene star over any path expression. Now, we can only use it on atomic edge labels, or inverse of edge labels. In particular, this implies that we cannot ask for the transitive closure of path expressions that interact with data.

We make the following observation:

Observation 33. *There is a function $f : U \times$ Reg-GXPath$^{pos}_{core} \to U \times$ Reg-GXPath^{pos} computable in polynomial time such that $[\![G]\!]_\eta = [\![G']\!]_{\eta'}$, G' has polynomial size on G, η' has polynomial size on η and η' does not use the $*$ operator.*

Proof. For every edge label $l \in \Sigma_e$, we create a new label l^* and add the edges (v, l^*, w) to G for each pair of nodes $v, w \in V_G$ such that $(v, w) \in [\![G]\!]_{\downarrow_l^*}$. This is our desired G'. To define η', we only need to change any subformula $\downarrow_l^*$ to $\downarrow_{l^*}$ in η. The same can be done for the subformulas of the form $\downarrow_l^{-*}$. □

Actually, this procedure may be utilized even when the Kleene star is applied to arbitrary path expressions α, by defining a new edge label $\downarrow_{\alpha^\star}$ and joining every pair of nodes v, w such that $(v, w) \in [\![\alpha^*]\!]$. However, the difference is that if we only use the $*$ operator in atomic expressions, then the fact that $(v, w) \in [\![\alpha^*]\!]$ is independent of the data values of the data-graph. Due to this fact, we deduce that:[7]

Lemma 34. *Given a Reg-GXPath^{pos} formula η that does not use the Kleene star, there is a constant c_η such that if $(v, w) \in [\![\eta]\!]_G$ for some data-graph G, then there is a subset of G with at most c_η nodes V_η such that $(v, w) \in [\![\eta]\!]_{G_{V_\eta}}$, where G_X is the subgraph of G induced by the nodes X.*

Proof. We prove this by induction on the formula's structure, in order to obtain a tight bound.

The base cases are trivial. This is, $c_\epsilon = 1$, $c_{\downarrow_l} = 2$, $c_{[\mathrm{D}^=]} = 1$, and so on.

For the inductive case, it is easy to see that $c_{\alpha.\beta} = c_\alpha + c_\beta - 1$, $c_{\alpha \cup \beta} = \min(c_\alpha, c_\beta)$, $c_{\alpha \cap \beta} = c_\alpha + c_\beta - 1$, and so on. The most interesting case is $\alpha^{n,m}$, where we have that $c_{\alpha^{n,m}} = m \times c_\alpha$. □

Observe that c_η might be exponential with respect to $|\eta|$, since in $\alpha^{n,m}$ the size of m is $\log(m)$.

Notice that, given a Reg-GXPath$^{pos}_{core}$ formula without the Kleene star, now we can obtain a bound for the number of nodes that interact with the formula by evaluating it at an origin. Hence, to decide whether a data-graph G satisfies a given formula $\eta \in$ Reg-GXPath$^{pos}_{core}$ from an origin o, it suffices to evaluate it in every data-graph $G' \subseteq G$ where $|V_{G'}| < |c_\eta|$ and $o \in V_{G'}$. It follows that, if the formula is satisfied by any G' then G will also satisfy it (due to the monotonicity of Reg-GXPath^{pos}, see Section 3).

We will show a polynomial restriction of the ISOMORPHIC-REPAIR problem, but first we need the following lemma:

Lemma 35. *Let A be a subset of n distinct data-values that do not appear in η. If a data-graph of n nodes satisfies η, then there is another data-graph that satisfies η and only uses data-values from $\Sigma_n^\eta \cup A$.*

Proof. The main idea behind this lemma is that the "identity" of those data values that are not in Σ_n^η is not important, since they only satisfy expressions of the form $[c]^{/=}$, $\langle \alpha \neq \beta \rangle$ or $\langle \alpha = \beta \rangle$. Therefore, if we modify them still preserving the equalities between them and the fact that they are not mentioned in η, the satisfiability of η is not affected. The details are in the Appendix 9. □

Based on Lemmas 34 and 35, we obtain the following result:

Theorem 36. *Let $\varphi \in$ Reg-GXPath$^{pos}_{core}$ be a fixed node expression that is evaluated in an origin, then the* ISOMORPHIC-REPAIR *problem can be solved in polynomial time.*

[7] It is enough to restrict the $*$ operator to be used only in subexpressions η that do not contain any data operation (i.e. a datatest $[\mathrm{c}^=]$ or $[\mathrm{c}^{\neq}]$ or a data comparison like $\langle \alpha = \beta \rangle$ or $\langle \alpha \neq \beta \rangle$).

Proof. Given the data-graph G and the expression φ, we start by removing the Kleene operator from φ as in Observation 33. Let φ' be this new formula, alongside our new data-graph G'. To decide whether there exists a data-graph H such that $H \equiv G'$ and $H, o \models \varphi$, it is enough to find a sub data-graph H' induced by some nodes of G' containing o with size less or equal to $c_{\varphi'}$ such that $H', o \models \varphi'$.

Notice that we only need to consider $\mathcal{O}(n^{c_{\varphi'}})$ such sub data-graphs. For each of these data-graphs, we should consider every possible assignment of data values to the nodes. Apart from those data values in Σ_n^φ, we might need some extra data values, since it could be the case that we need to satisfy some subformula of the form $\bigwedge_{\mathrm{C} \in \Sigma_n^\varphi} \mathrm{C}^{\neq}$; or also an inequality of data values such as $\langle \alpha_1 \neq \alpha_2 \rangle$.

By Lemma 35, it suffices to choose a set A of c_φ arbitrary data values not mentioned in φ. Then, for every sub data-graph H' of at most c_φ nodes we need to consider each data value assignment of its nodes that only uses data values from $\varphi \cup A$. We can bound the number of such assignments by $(|\varphi| + c_\varphi)^{c_\varphi}$. Since φ is fixed, this bound is $\mathcal{O}(1)$. Finally, noticing that checking if a certain data-graph satisfies a given node expression at an origin o can be done in polynomial time on the data-graph and formula size, we conclude that the algorithm that checks every possible data value assignment for each possible sub data-graph H' of at most c_φ nodes of G' runs in poly($|G|$). □

Now we focus our attention on the DATA CLEANING problem for the particular relaxation in which the ISOMORPHIC-REPAIR is tractable. First, we state a lemma that will be useful in the proof of the main result, which is then presented in Theorem 38.

Lemma 37. *Let $G = (V_G, L_G, D_G)$ be a data-graph of n nodes with an origin o, let φ be a node expression from Reg-GXPath$^{pos}_{core}$ that does not contain any formula $\langle \alpha = \beta \rangle$ as subexpression, and let $\delta : \Sigma_n \times \Sigma_n \to \mathbb{N}_0$ be a function such that, for each fixed $\mathrm{C} \in \Sigma_n$, we can efficiently enumerate the set $\mathrm{D}_1, \mathrm{D}_2, \ldots$ where $\delta(\mathrm{D}_i, \mathrm{C}) \leq \delta(\mathrm{D}_{i+1}, \mathrm{C})$, and every data value D belongs to the enumeration.*

If there is a data-graph $H = (V_G, L_G, D_H)$ that satisfies φ and minimizes $\sum_{v \in V_G} \delta(D_H(v), D_G(v))$, then there is another data-graph $H' = (V_G, L_G, D_{H'})$ that fulfills the same properties as H and such that, for each data value C in G that was modified in H' to some data value D that is not mentioned in φ, satisfies that $\mathrm{D} = \mathrm{D}_j$ for $j \leq n + |\Sigma_n^\varphi|$ where D_i is the ith data value in the efficient enumeration obtained for C with respect to δ.

Proof. The main idea is that those nodes that are mapped to data values outside the set Σ_n^φ can be remapped to others (and preferably those that appear first in the enumeration $\mathrm{D}_1, \mathrm{D}_2, \ldots$), as long as this does not provoke that a subexpression of the form $\langle \alpha \neq \beta \rangle$ is not satisfied. We can guarantee that this will not happen by choosing a data value between the first $n + |\Sigma_n^\varphi|$, since there will always be some data value here that is not used in the data-graph. The details are in the Appendix 9. □

Theorem 38. *The* DATA CLEANING *problem can be solved in polynomial time when considering a PUDG $(\mathcal{I}, \mathcal{R}, G)$ and an origin o such that:*

- *$\mathcal{I}$ depends on a fixed node expression $\varphi \in$ Reg-GXPath$^{pos}_{core}$ that does not contain a data comparison with equality (i.e., the formula $\langle \alpha = \beta \rangle$) as a subexpression, and such that $\mathcal{I}(H) > 0$ if and only if $H, o \models \varphi$ and $\mathcal{I}(H) = \mathcal{I}(H')$ if $H, o \models \varphi$ and $H', o \models \varphi$.*[8]
- *$\mathcal{R}$ depends on a local function δ, as those considered in Section 5.2, such that, given a data value C we can efficiently enumerate all data values $\mathrm{D}_1, \mathrm{D}_2, \mathrm{D}_3 \ldots$ so that $\delta(\mathrm{D}_i, \mathrm{C}) \leq \delta(\mathrm{D}_{i+1}, \mathrm{C})$.*

Proof. We will assume that φ does not use the Kleene star operator, otherwise we apply first the transformation of Observation 33.

We need to find a data-graph H such that $H, o \models \varphi$, and H minimizes the cost of the data modifications between H and G. Naturally, we only need to modify the data values of the sub data-graph that satisfies

[8] At least one data-graph H must exist for this distribution to be well defined.

φ, since leaving the remaining data values as they are is the most inexpensive strategy[9]. To do this, we compute the cost of the transition for every sub data-graph and every assignment of data values in the algorithm given in Theorem 36. Notice that, if any set of nodes is mapped to a set of data values that are not mentioned in φ, then we use the property imposed for δ to compute the most inexpensive data values from $\Sigma_n \setminus \Sigma_n^\varphi$.

It follows from Lemma 34 that there is a data-graph that coincides with $G = (V_G, L_G, D_G)$ except for its data values, and satisfies φ if and only if there is a subset $V_\varphi \subseteq V(G)$ such that $|V_\varphi| = c_\varphi$, $o \in V_\varphi$, and the sub data-graph induced by V_φ satisfies φ, where some of the data-values may have changed. Notice that there are at most $\binom{n}{c_\varphi} = \mathcal{O}(n^{c_\varphi})$ such subsets, which is polynomial on the input size.

As proposed in Theorem 36, we could iterate on each of these subsets and consider every possible assignment of data values that only uses data values from Σ_n^φ or an arbitrarily chosen set A of data values that is not mentioned in φ with $|A| = c_\varphi$. However, in this case, such set A might not contain the optimum transitions based on the transitions costs defined by the function δ. Nonetheless, we can use Lemma 37 to show that there is a bounded number of assignments that we need to consider in order to find the optimum. Therefore, we can adapt the algorithm of Theorem 36 to consider a specific set of data values assignments that we have shown contain an optimum assignment over δ and also satisfies φ (if any such assignment exists). Observe that there are $|\Sigma_n^\varphi| + 2c_\varphi$ options for each node, thus there is a total of $(|\Sigma_n^\varphi| + 2c_\varphi)^{c_\varphi}$ assignments to consider. Finally, we keep the best assignment, which requires $\mathcal{O}(n^{c_\varphi})$ operations. □

Theorem 38 captures a restriction of the problem in which Reg-GXPath$_{core}^{pos}$ expressions are allowed (as long as they do not contain the data equality operator $\langle \alpha = \beta \rangle$) and the data cleaning problem can be solved in polynomial time. We can compare the constraints expressible through the fragment described in Theorem 38 to the ones studied in other contexts where path constraints are evaluated in an origin, such as [2, 18, 11]. In comparison with those works, Reg-GXPath$_{core}^{pos}$ allows to define more complex paths, including subpaths through the nesting operator $\langle \alpha \rangle$ (which has many applications in the context of the Resource Description Framework [13]), and data tests alongside data comparison for inequality. Both features are not available in the commonly studied Regular Path Constraints (RPCs), since they do not interact with the data values in the nodes. Nevertheless, RPCs allow to express non-monotonic properties through path implications (i.e. if a path α starts in o then another path β should also start in o), which cannot be simulated in our context. It would be interesting to study if it is possible to define implication-like expressions via Reg-GXPath$_{core}^{pos}$ constraints, while allowing the data cleaning problem to remain solvable in polynomial time.

Example 39. To exemplify this restriction, imagine having a graph database storing geographical data, such as airports and train stations, in the form of nodes. The edge labels indicate ways to move between them, related to different types of transportation: TRAIN or PLANE. Furthermore, simulating some features of the Resource Description Framework, we could imagine that each location node has a TYPE edge pointing to a metadata node that contains some extra information, such as whether the location is the MAIN one of the state.

A particular airport wants to ensure that every airport of a certain list is reachable using at most two transfers (i.e. at most three different transports) through MAIN stations. Such airport can be considered the origin of the data-graph, and we can write the following rule

$$\varphi = \langle \bigwedge_{\text{AIRPORT\_NAME} \in \text{AIRPORTS\_LIST}} (((\downarrow_{\text{TRAIN}} \mid \downarrow_{\text{PLANE}})\langle \downarrow_{\text{TYPE}} [\text{MAIN}^{=}] \rangle)^{0,3} [\text{AIRPORT\_NAME}^{=}] \rangle$$

to capture this condition. Then, if this restriction does not hold due to data pollution on the nodes (such as improper data entry tasks or non-standard naming of the airports), we would perform a data cleaning task considering a $\mathcal{I}$ depending on φ, and a $\mathcal{R}$ depending on a local function δ modelling string perturbances, such as the Levenshtein distance. Moreover, such local function δ could be defined to prevent altering the metadata nodes, if desired.

[9]Notice that this property of local functions as defined in Section 5.2 is not really needed for this algorithm.

6.2 Adding soft and hard constraints

While a particular EMDG $(\mathcal{I}, \mathcal{R})$ can be used to encode information about our epistemic understanding, sometimes it makes sense to restrict our space of possible worlds in particular ways. Depending on the field of application, these restrictions could arise from obtaining a deeper theoretical understanding, as a way to explore multiple hypotheses or from practical limitations on the world-models that can be effectively analyzed.

The concept of hard and soft constraints from the area of database repairing [11, 47, 7] can be incorporated to our framework of data-graphs, for both problems of Data Cleaning and PQA.

Definition 40 (Restrictions and consistency)**.** Let G be a data-graph and $\mathcal{C} = \mathcal{P} \cup \mathcal{N}$ a set of **restrictions** (also called **constraints**), where $\mathcal{P}$ consists of path expressions and $\mathcal{N}$ of node expressions. We say that G **violates** φ, if $x \notin [\![\varphi]\!]_G$ for some $x \in V_G$. Otherwise, we say that G **satisfies** φ, and we denote it by $\boldsymbol{G \models \varphi}$. Similarly, we say that G **violates** α if $(x, y) \notin [\![\alpha]\!]_G$ for some $x, y \in V_G$, and otherwise we say that G **satisfies** α ($\boldsymbol{G \models \alpha}$).

If G does not violate η for any $\eta \in \mathcal{C}$, we say that G is **consistent** w.r.t. $\mathcal{C}$, and we note this as $G \models \mathcal{C}$. We say that G, x **violates** φ, if $x \notin [\![\varphi]\!]_G$. Similarly, we say that G, x, y **violates** α if $(x, y) \notin [\![\alpha]\!]_G$.

This notion of consistency is useful for dividing the set of data-graphs into two natural groups: those who satisfy the constraints and those who do not. However, this stark division can be generalized.

Given a set of constraints $\mathcal{C}$, we can assign a weight $0 \leq w_\eta < 1$ to each $\eta \in \mathcal{C}$ by considering a function $w : \mathcal{C} \to [0, 1)$. A value of 0 can be thought of as a **hard constraint**: we completely exclude data-graphs which violate this formula. On the other hand, a value strictly between 0 and 1 is a **soft constraint**: the models that violate this formula are penalized in their probability, but not absolutely.

A weight function w defines a preference relation among constraints. Given $(\mathcal{I}, \mathcal{R})$ an EMDG, we would like to lift this preference relation to a preference relation among the data-graphs G in $\mathcal{I}$, combining the original probability of the data-graph with the weights from each constraint in $\mathcal{C}$ that is satisfied by G. We formalize this idea as follows:

$$w(G) \stackrel{\text{def}}{=} \mathcal{I}(G) \times \prod_{\eta \in \mathcal{C}} v(G, \eta)$$

where $v(G, \eta)$ is defined as:

$$v(G, \eta) \stackrel{\text{def}}{=} \begin{cases} w(\eta) & \text{if } G \not\models \eta \\ 1 & \text{if } G \models \eta \end{cases}$$

Given $(\mathcal{I}, \mathcal{R})$ an EMDG, we can now define the new EMDG $(\mathcal{I}_w, \mathcal{R})$ which is an EMDG where the weights given by w have been incorporated and normalized from the original values of $\mathcal{I}$.

$$\mathcal{I}_w(G) \stackrel{\text{def}}{=} \frac{w(G)}{\sum_{H \in \mathcal{I}} w(H)}$$

Notice that incorporating w does not introduce modifications in $\mathcal{R}$, but it does change $\mathcal{B}$ (see (1) for the formal definition of $\mathcal{B}$).

Complexity considerations Given an EMDG $(\mathcal{I}, \mathcal{R})$ and a weighted restriction set w, how much harder are the problems of Data cleaning and PQA over $\mathcal{I}_w$? Observe that, since $\mathcal{I}$ could contain an infinite number of data-graphs of positive probability, computing $\mathcal{I}_w$ explicitly might not be possible.

Example 41. Consider a simple case where $\mathcal{C}$ consists of a single node or a single path expression, that is, $\mathcal{C} = \{\eta\}$. Furthermore, assume that we know the total probability mass p_η of data-graphs in $\mathcal{I}$ where η

holds. Then it follows from a simple calculation that:

$$\mathcal{I}_w(G) = \frac{\mathcal{I}(G) \times v(G,\eta)}{p_\eta + w(\eta)\times(1-p_\eta)}$$

Indeed, this is a consequence of the definition of $\mathcal{I}_w(G)$ and rewriting the denominator as follows:

$$\begin{aligned}
\sum_{H\in\mathcal{I}} w(H) &= \sum_{H\in\mathcal{I}\,|\,H\models\eta} w(H) + \sum_{H\in\mathcal{I}\,|\,H\not\models\eta} w(H) \\
&= \sum_{H\in\mathcal{I}\,|\,H\models\eta} (\mathcal{I}(H)\times v(H,\eta)) + \sum_{H\in\mathcal{I}\,|\,H\not\models\eta} (\mathcal{I}(H)\times v(H,\eta)) \\
&= \sum_{H\in\mathcal{I}\,|\,H\models\eta} (\mathcal{I}(H)\times 1) + \sum_{H\in\mathcal{I}\,|\,H\not\models\eta} (\mathcal{I}(H)\times w(\eta)) \\
&= \sum_{H\in\mathcal{I}\,|\,H\models\eta} \mathcal{I}(H) + \left(w(\eta) \times \sum_{H\in\mathcal{I}\,|\,H\not\models\eta} \mathcal{I}(H) \right) \\
&= p_\eta + w(\eta)\times(1-p_\eta)
\end{aligned}$$

Notice that the same idea applies for more complex sets of constraints, as long as we know the probability masses in $\mathcal{I}$ for each combination of Boolean values of the formulas in $\mathcal{C}$ (e.g., if $\mathcal{C} = \{\varphi,\psi\}$, we ask for $p_{\varphi,\psi}, p_{\neg\varphi,\psi}, p_{\varphi,\neg\psi}, p_{\neg\varphi,\neg\psi}$, respectively meaning the total probability of data-graphs satisfying the two formulas in the sub-index).

Example 42. Consider the example depicted in Figure 4. A reasonable integrity constraint for some social networks is that of symmetry for the friendship relation. If we want to exclude data-graphs where this condition does not hold everywhere, then the restriction can be expressed in Reg-GXPath via $\alpha = {\downarrow_{\text{FRIEND}}} \Rrightarrow {\downarrow^{-}_{\text{FRIEND}}}$, and applied as the sole hard constraint ($w(\alpha) = 0$) as a way to exclude those data-graphs which violate it.

Over a pair of nodes, the satisfaction of this path expression indicates that, if the first node is friends with the second one, then the second one must also be friends with the first one. The violation of this restriction over a data-graph means that there is a pair of nodes where this does not hold, and thus that the friendship relation is not symmetric in that data-graph.

Now, if we modify the original $\mathcal{I}$ given in the aforementioned figure, then the middle and lower graphs on the left side (G_2 and G_3) would have probability 0 in the modified probabilistic database $\mathcal{I}_w$ (and thus the function $\mathcal{B}$ would now assign probability 1 to the upper left graph when starting from G').

7 Probabilistic query answering

In this section, we focus on the problem of probability query answering for PUDGs. We first define the problem in general terms and later study the complexity for the case of subset and superset PUDG, respectively.

Definition 43. Given $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$ and a node or path expression η, we define **the probability of η over $\mathcal{U}$** (or over $\mathcal{I}$ given G' and $\mathcal{R}$) as:

$$\sum_{G\in\mathcal{I}} A_{G,\eta} \times \mathcal{B}(G')(G),$$

where $A_{G,\eta}$ is the indicator function of "η holds over all nodes (or pairs of nodes) in G".

Definition 44 (`Global PQA` over data-graphs)**.** We define the probabilistic query answering problem for graph-databases:

PROBLEM: The probability that a node [path] query holds over all nodes [pairs of nodes]. INPUT: A PUDG $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$ and a node [path] query η OUTPUT: The probability of η over $\mathcal{I}$, given G' and $\mathcal{R}$.

Definition 45 (`Existential PQA` over data-graphs)**.** Similarly, we could define the probabilistic query answering for graph-databases:

PROBLEM: The probability that a node [path] query holds over *at least* one node [pair of nodes]. INPUT: A PUDG $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$ a node [path] query η OUTPUT: The probability that η holds over at least one node [pair of nodes] in the data-graphs from $\mathcal{I}$, given G' and $\mathcal{R}$.

However, notice that this problem is the dual of global PQA: given a node expression φ [a path expression α], the answer to the Global PQA problem for that formula coincides with 1 minus the answer to the Existential PQA problem for $\neg\varphi$ [$\overline{\alpha}$], and vice versa:

Let $A_{G,\varphi}$ be defined as in Definition 43 for node expression φ, and $E_{G,\varphi}$ be the binary value of "φ holds over some node in G", we have:

$$\begin{aligned}\sum_{G\in\mathcal{I}} A_{G,\varphi} \times \mathcal{B}(G')(G) &= \sum_{G\in\mathcal{I}} (1 - E_{G,\neg\varphi}) \times \mathcal{B}(G')(G) \\ &= \sum_{G\in\mathcal{I}} \mathcal{B}(G')(G) - \sum_{G\in\mathcal{I}} E_{G,\neg\varphi} \times \mathcal{B}(G')(G)\end{aligned}$$

And therefore:

$$\sum_{G\in\mathcal{I}} A_{G,\varphi} \times \mathcal{B}(G')(G) = 1 - \sum_{G\in\mathcal{I}} E_{G,\neg\varphi} \times \mathcal{B}(G')(G).$$

We also define a decision problem related to `Global PQA`, namely GLOBAL PQA BOUND:

PROBLEM: Global PQA bound. INPUT: A PUDG $\mathcal{U} = (\mathcal{I}, \mathcal{R}, G')$, a node [path] query η, and bound b. OUTPUT: Decide if the probability of η over $\mathcal{I}$, given G' and $\mathcal{R}$, is above b.

Example 46 (Global PQA—probability that an (incomplete) observed network has no point of attack)**.** Suppose that an attacker observes a data-graph G' representing an incomplete diagram of a computer network. Assume they want to query whether all the nodes of the (full) network are secure, assuming for simplicity that this is represented by them having the data value SECURE, and they have a model of possible networks ($\mathcal{I}$) and of the limitations of their data-gathering methods ($\mathcal{R}$).

In this case, it makes sense for the semantics of the probability in the PQA to represent the proportion of clean data-graphs where *all* nodes are secure, i.e., the (weighted) proportion of $G \in \mathcal{I}$ such that for all $x \in G$ we have that $G, x \models \text{SECURE}^{=}$. The relative proportion of secure or not secure nodes should not be reflected in the calculated probability: the only thing that matters is the existence of at least a single point of attack. In other words, we are talking of the global PQA problem.

7.1 PQA for Subset PUDG

As in Section 4, first we study the complexity of the PQA problem for subset PUDGs.

One way to solve the Global PQA problem is to evaluate the query η in every graph in the cosupport of G' and then sum all the results multiplied by the probabilities associated. If $\sigma(G')$ is polynomially bounded on G', then this approach would be feasible. Therefore, based on our results for data cleaning, namely Theorem 19 for subset and Theorem 21 for superset, we conclude that:

Theorem 47. *The Global PQA problem can be solved in polynomial time in Subset PUDGs if the number of nodes and edges deleted from the original graph, as well as the number of possible data-values that can be assigned to a node, are bounded by some constant.*

Let us consider the simplified case in which no node deletions are allowed, and therefore the only possible data-graphs in the cosupport of G' are graphs obtained by adding edges to G'. We can prove the following bounds:

Theorem 48. *There exists a Reg-GXPath formula η such that the problem Global PQA is PP-hard*[10] *when considering no node deletions through $\mathcal{R}$. Moreover, PSPACE is an upper bound for the problem.*

Proof. For the first statement we will reduce MAJSAT (*majority SAT*) to subset PQA where no node deletions are allowed. MAJSAT problem consists in, given a boolean formula φ with free variables, deciding if more than half of the assignments of the free variables of φ satisfy it. We can assume that φ is given in conjunctive normal form.

Let $x_1, ..., x_n$ be the free variables of φ, and let $c_1, ..., c_m$ be its clauses. We will construct an instance of the subset PQA problem such that half of the assignments of φ satisfy it if and only if the answer of the PQA problem is greater than $\frac{1}{2}$.

We define the observed data-graph G' in the following way. Let $V'_G = \{v_i \mid 1 \leq i \leq n\} \cup \{w_j \mid 1 \leq j \leq m\} \cup \{\top, \bot\}$, where the nodes v_i refer to the variables x_i and the nodes w_j refer to the clauses c_j. The edges of the graph are related to the structure of the formula φ as follows: $L(v_i, w_j) = \{\downarrow_+\}$ if the variable x_i appears in the clause c_j without negation, and $L(v_i, w_j) = \{\downarrow_-\}$ if the variable x_i appears with negation in c_j. For every other pair of nodes v, w that do not satisfy any of those hypotheses, we define $L(v, w) = \emptyset$. Also, we set $D(w_j) = clause$, $D(v_i) = var$ and $D(\star) = \star$ for $\star \in \{\top, \bot\}$. Observe that the data-graph G' contains enough information to rebuild φ.

The image of $\mathcal{B}$ will represent the set of all possible assignments of the variables $x_1, ..., x_n$. To achieve this, we define $\mathcal{I}(H) = \frac{1}{2^n}$ if $G' \subseteq H$, every node v_i has an edge $\downarrow_{\text{ASSIGNED}}$ to either $\bot$ or $\top$ (but not to both at the same time), and if no other edge or node was added to obtain H from G'. We can easily map the data-graph H to an assignment of the variables x_i by defining that x_i is valuated to true if and only if the edge $(v_i, \text{ASSIGNED}, \top)$ belongs to H. We can also do this in the other way: given an assignment v of the variables of φ we can build the data-graph H_v that corresponds to that assignment.

We also define $\mathcal{R}(G)(G') = 1$ for every graph G.

Finally, the query η will be in charge of 'checking' whether the assignment represented by any $H \in \mathcal{B}(G')$ satisfies the Boolean formula φ. This can be captured by defining:

$$\eta = [clause]^{=} \Rightarrow \langle \downarrow_+^- \downarrow_{\text{ASSIGNED}} [\top]^{=} \rangle \vee \langle \downarrow_-^- \downarrow_{\text{ASSIGNED}} [\bot]^{=} \rangle$$

Now we prove the following fact: *a valuation v over the variables of φ satisfies φ if and only if the data-graph $G' \subseteq H_v$ that represents such assignment v satisfies the constraint η.*

Let v be an assignment of the variables of φ, and let H_v be the data-graph that corresponds to that assignation. That is, $H_v = (V_{H_v}, L_{H_v}, D_{H_v})$ is defined as:

$$V_{H_v} = V_{G'}$$
$$D_{H_v}(z) = D_{G'}(z) \text{ for } z \in V_{H_v}$$
$$L_{H_v}(v_i, \star) = \{\downarrow_{\text{ASSIGNED}}\} \text{ if } v(x_i) = \star \text{ for } \star \in \{\top, \bot\}. \text{ Otherwise } L_{H_v}(v_i, \star) = \emptyset$$
$$L_{H_v}(z_1, z_2) = L_{G'}(z_1, z_2) \text{ for any other case}$$

Suppose that v satisfies φ and let z be a node of H_v. If $D_{H_v}(z) \neq clause$ then $z \in [\![\eta]\!]_{H_v}$ by definition of η. Otherwise, z is actually w_j for some j. Observe that the clause c_j is satisfied by v, and therefore there is a variable x_i that either appears without negation on c_j and is valuated to true by v, or either

[10] PP [41] is the class of decision problems solvable by a probabilistic Turing machine in polynomial time, with an error probability of less than $\frac{1}{2}$ for every instance. MAJSAT is a well-known PP-hard problem.

appears negated in c_j and is valuated to false by v. In the first case, there must be edges $(v_i, +, w_j)$ and $(v_i, \text{ASSIGNED}, \top)$ in the graph, while in the other case there will be edges $(v_i, -, w_j)$ and $(v_i, \text{ASSIGNED}, \bot)$. In either case, $c_j \in [\![\eta]\!]_{H_v}$ since it satisfies the consequent of η.

Now for the other direction, suppose that v does not satisfy φ. Let c_j be the clause that is not satisfied. Then, the node w_j would not belong to $[\![\eta]\!]_{H_v}$: if there is an edge $(v_i, +, c_j)$, then there will be no edge $(v_i, \text{ASSIGNED}, \top)$ in H_v, since that would imply that $v(x_i) = \top$ and c_j is satisfied. The same reasoning can be followed for the $(v_i, -, c_j)$ edges.

To complete the reduction, suppose that $\varphi \in MAJSAT$. Thus, let $\mathcal{V}$ be the set of at least $2^{n-1} + 1$ assignments of φ that satisfy it. Observe that every data-graph of the set $\{H_v\}_{v \in \mathcal{V}}$ satisfies η, and also that:

$$\mathcal{B}(G')(H_v) = \frac{\mathcal{I}(H_v) \times \mathcal{R}(H_v)(G')}{\sum_{I \in \mathcal{I}} \mathcal{I}(I) \times \mathcal{R}(I)(G')} = \frac{\mathcal{I}(H_v)}{\sum_{I \in \mathcal{I}} \mathcal{I}(I)} = \mathcal{I}(H_v) = \frac{1}{2^n}.$$

Then, the probability that η holds over $\mathcal{I}$ given G' and $\mathcal{R}$ must be equal or greater than $|\mathcal{V}| \times \frac{1}{2^n} \geq \frac{2^{n-1}+1}{2^n} > \frac{1}{2}$.

For the other direction, if $\varphi \notin MAJSAT$, then half or more of its assignments do not satisfy it. It follows that at least half of the data-graphs in $\mathcal{I}$ do not satisfy η, because there is a bijection between the data-graphs $H \in \mathcal{I}$ and the valuations over φ, and this bijection relates formulas that satisfy φ with data-graphs that satisfy η. Since for every data-graph $H \in \mathcal{I}$ holds that $\mathcal{B}(G')(H) = \frac{1}{2^n}$, it follows that the probability that η holds over $\mathcal{I}$ given G' and $\mathcal{R}$ is bounded by $\frac{1}{2}$.

For the upper bound, we show that the PQA subset problem can be solved in PSPACE. Notice that any data-graph H with the same number of nodes as the observed data-graph G' can be represented by a binary string of length $n^2 \times |\Sigma_e|$ that defines which edges belong to H. One way to solve the problem would be to iterate over this 'counter' and, for each representation ω, to construct the data-graph H and check whether the formula η is satisfied in H. We also use an accumulator where we sum the values $\mathcal{B}(G')(H)$ of those graphs that satisfy η. Since for every H the value $\mathcal{B}(G')(H)$ can be represented in polynomial size on $|G'|$, this accumulator also has polynomial size on G'[11]. All of this can be done in polynomial space, and requires exponential time. □

Observe that we could have used the node expression

$$\psi = [var]^{=} \vee [\top]^{=} \vee [\bot]^{=} \vee \langle \downarrow_{+}^{-} \downarrow_{\text{ASSIGNED}} [\top]^{=} \rangle \vee \langle \downarrow_{-}^{-} \downarrow_{\text{ASSIGNED}} [\bot]^{=} \rangle$$

to obtain the same bound but using an expression from Reg-GXPath^{pos}, which is a weaker language.

7.2 PQA for Superset PUDG

We finish this section by giving complexity considerations for PQA in superset PUDGs. In this case, the PQA problem can always be solved in polynomial space if $\mathcal{I}$ and $\mathcal{R}$ can be computed efficiently[12]: a PSPACE algorithm would just iterate over every graph $G \subseteq G'$ and accumulate the value $A_{G_i,\eta} \times \mathcal{B}(G')(G)$. Observe that the normalization factor of $\mathcal{B}$ can also be computed.

This is an improvement from the subset case: the general PQA subset problem was not solvable in PSPACE because we had no way of knowing whether the graphs generated while iterating over the cosupport of G' were bounded by some polynomial. In this case, since I is in the cosupport, we can guarantee that $I \subseteq G'$.

Nevertheless, the problem remains hard even in restricted scenarios:

[11] One way to prove this fact is by noticing that the number of bits that the counter needs to represent its current value will never be more than those needed by the value $\mathcal{B}(G')(H)$ with the biggest number of bits used. This number of bits can be bounded by a polynomial on G', and therefore the number of bits used by the accumulator will always be bounded by this polynomial, since the sums will never require an extra bit to represent the output.

[12] In fact we only require them to use polynomial space to compute its outputs, which as well must be bounded by a polynomial.

Theorem 49. *The PQA superset problem is* PP-HARD *when considering no node additions through* $\mathcal{R}$.

Proof. The proof is rather similar to the subset case. Given an input formula φ with n variables x_i and m clauses c_j for the MAJSAT problem, we define the graph $G' = (V, L, D)$ as follows:

$$
\begin{aligned}
&V = \{v_i : 1 \leq i \leq n\} \cup \{w_j : 1 \leq j \leq m\} \cup \{\top, \bot\}\\
&L(v_i, w_j) = \{\downarrow_+\} \iff x_i \text{ appears without negation in } c_j\\
&L(v_i, w_j) = \{\downarrow_-\} \iff x_i \text{ appears with negation in } c_j\\
&L(v_i, \top) = \{\downarrow_{\text{ASSIGNED}}\} \text{ for every } 1 \leq i \leq n\\
&L(v_i, \bot) = \{\downarrow_{\text{ASSIGNED}}\} \text{ for every } 1 \leq i \leq n\\
&L(z_1, z_2) = \emptyset \text{ for any other case}\\
&D(c_i) = \text{CLAUSE}\\
&D(x_i) = \text{VAR}\\
&D(\star) = \star \text{ for } \star \in \{\top, \bot\}
\end{aligned}
$$

We define $\mathcal{R}(I)(G') = 1$ for every $I \subseteq G'$, and $\mathcal{I}(I) = \frac{1}{2^n}$ if and only if I represents an assignment of the variables x_i, following the semantics defined in the previous proof.

Using the same Reg-GXPath expression η as in the subset case, it is easy to prove that φ is satisfied by more than half of its assignments if and only if the answer to superset PQA problem with inputs G', $\mathcal{I}$, $\mathcal{R}$ and $\frac{1}{2}$ is 1. □

8 Conclusions and future work

In this work, we developed a formal framework for probabilistic unclean data-graphs, which allowed us to study several basic problems within them. The formalism defined in Section 3 expands the ideas described in [44], addressing particular difficulties that arise when dealing with data-graphs. We also described a set of restrictions over the initial model that are analogous to those found in other works of the area, mainly the usual subset and superset restrictions considered when dealing with inconsistent databases as in [16] or [11], and we show that they still hold a reasonable expressive power to capture real-life examples. We defined the problems of DATA CLEANING and PROBABILISTIC QUERY ANSWERING for unclean data-graphs, which strongly relate with two fundamental questions in the context of unclean databases developed in [44]. First, given an observed database and a model for the evolution of inconsistencies: which is the most likely original state of the data-graph? And second, given the same prior knowledge: which is the probability that a certain condition or constraint holds in the original state of the data-graph? The results for both problems are summarized in Table 2 and Table 3.

In Section 4 and Section 7 we studied specific restrictions of the problems and determined that the complexity of the subset and superset versions of DATA CLEANING and PROBABILISTIC QUERY ANSWERING is hard in both cases. Nevertheless, we provided additional restrictions for which the problem becomes tractable. Regarding the study of PQA in probabilistic data-graphs, we considered GLOBAL PQA and EXISTENTIAL PQA over data-graphs, and studied this using the Reg-GXPath syntax to express conditions or constraints over data-graphs. It follows from the proofs given in Section 7 that the results obtained can be generalized to similar graph query languages, as long as they are (1) expressive enough to define simple path conditions on every pair of nodes and (2) simple enough to compute the set of answers given a data-graph instance in polynomial time.

Following a similar idea, we defined the *update PUDG* restriction over the PUDG model, in which we consider the possibility of modification of labels and data values in edges and nodes, respectively. This resembles an analogous restriction already studied in the literature of inconsistency reasoning: namely, attribute-based repairs [15]. In the update case, we obtained intractable results for quite restrictive scenarios,

Π PUDG	Restrictions on $\mathcal{I}$	Restrictions on ICs	Version	Complexity	Reference
subset	–	–	D	NP-c	Th. 18
subset	bounded edge additions	–	F	poly	Th. 19
superset	–	–	D	NP-c	Th. 20
superset	bounded node/edge additions	–	F	poly	Th. 21
update	–	–	D	NP-c	Th. 22
node update	k-data-prior function	–	F	poly	Th. 23
node update	global cardinality	–	F	poly	Th. 24
node update	weak cardinality	–	F	poly	Th. 25
node update	valid data preselection	–	F	NP-hard	Th. 26
node update	edge-label preselection	–	F	NP-hard	Th. 27
node update	–	$^{(*)}\text{Reg-GXPath}^{pos}_{core}$	F	poly	Th. 38

$^{(*)}$ $\text{Reg-GXPath}^{pos}_{core}$ without $\langle \alpha = \beta \rangle$

Table 2: Summary of the most notable results obtained for Data Cleaning. Column 'Version' states whether we are dealing with the decision (D) or the functional (F) version of the corresponding problem. In columns 'Restrictions on...' we refer to the most outstanding restrictions used for the particular problem; the complete extent of each result can be found in the corresponding theorem, specified in column 'Reference'.

which suggests that this problem is more difficult to address in the general case than those studied in Sections 4 and 7.

We observed in Section 6 that the concept of soft and hard constraints from the area of database repairing can be adapted to our framework of EMDGs and PUDGs. We do so by defining (weighted) restriction sets, which modify the distribution of a probabilistic data-graph by reducing the probability of those data-graphs which violate the constraints in the restriction set. A deeper complexity analysis for constraint satisfaction is a matter of our future work.

For the PQA problem when considering only subset PUDGs, the best upper bound we show is PSPACE, while the lower bound is PP, so there might be some room for improvement. In addition, we did not find restrictions over the realization model $\mathcal{R}$ that could make the problems easy, without trivializing it. In most cases, the interaction between really simple probabilistic databases $\mathcal{I}$ and realization models $\mathcal{R}$ already makes the problems hard, so one possible way to advance would be to consider instances of the Data Cleaning and PQA problem where the distribution defined by $\mathcal{I}$ is uniform and the weight of the reasoning is encoded in $\mathcal{R}$.

Π PUDG	Restrictions on ICs	Version	Complexity	Reference
subset	Reg-GXPath	F	poly	Th. 47
subset	(fixed) Reg-GXPath^{pos}	D	PP-hard	Th. 48
subset	Reg-GXPath	D	PSPACE	Th. 48
superset	Reg-GXPath	D	PP-hard	Th. 49

Table 3: Summary of the most notable results obtained for PQA. Column 'Version' states whether we are dealing with the decision (D) or the functional (F) version of the corresponding problem. In column 'Restricted on ICs' we refer to the most outstanding restrictions to the set of constraints considered in each case; the complete extent of each result can be found in the corresponding theorem, specified in column 'Reference'.

Regarding the expressiveness of the query language, subsets of Reg-GXPath could be explored to lower the complexity of the PQA problems. Nonetheless, observe that the results of hardness use only a narrow set of tools from the Reg-GXPath grammar, and therefore serious limitations on the complexity of the expression should be imposed. Considering expressions from weaker navigation languages such as those mentioned in [12] might be a good place to start. We also left open for exploration the possibility of semantic restrictions in the universe U of data-graphs, such as considering only data-trees, DAGs, or other type of structures, in order to investigate whether these restrictions provide a lowering of the complexity

for our problems. Similarly, we can study the trade-offs involved in further restrictions on the probabilistic data-graphs $\mathcal{I}$ and the noisy observers $\mathcal{R}$. Finally, it would also be interesting to consider an alternative but natural definition of the semantics of a given formula η over a data-graph, such that it returns the proportion of nodes or pairs of nodes that satisfy the formula. This direction is related to the possibility of extending our framework and definitions for studying open queries in PQA.

Acknowledgments

This work was funded in part by Universidad de Buenos Aires under UBACyT 20020190200124BA, CONICET under the PIP (grant 11220200101408CO), Agencia Nacional de Promoción Científica y Tecnológica, Argentina under grants PICT-2020-SERIEA-01481.

References

[1] Serge Abiteboul, T-H Hubert Chan, Evgeny Kharlamov, Werner Nutt, and Pierre Senellart. Capturing continuous data and answering aggregate queries in probabilistic XML. *ACM Transactions on Database Systems (TODS)*, 36(4):1–45, 2011. (Cited on 4)

[2] Serge Abiteboul and Victor Vianu. Regular path queries with constraints. *Journal of Computer and System Sciences*, 58(3):428–452, 1999. (Cited on 2, 22, 27)

[3] Foto N Afrati and Phokion G Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In *Proceedings of the 12th International Conference on Database Theory*, pages 31–41, 2009. (Cited on 10)

[4] Antoine Amarilli, Mikaël Monet, and Pierre Senellart. Conjunctive queries on probabilistic graphs: Combined complexity. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 217–232, 2017. (Cited on 4)

[5] Manish Kumar Anand, Shawn Bowers, and Bertram Ludäscher. Techniques for efficiently querying scientific workflow provenance graphs. In *EDBT*, volume 10, pages 287–298, 2010. (Cited on 1)

[6] Marcelo Arenas, Pablo Barceló, and Mikaël Monet. Counting problems over incomplete databases. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 165–177, 2020. (Cited on 4)

[7] Marcelo Arenas, Leopoldo Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *PODS*, volume 99, pages 68–79. Citeseer, 1999. (Cited on 2, 3, 28)

[8] Marcelo Arenas, Claudio Gutierrez, and Juan F. Sequeda. Querying in the age of graph databases and knowledge graphs. In *Proc. of the International Conference on Management of Data*, SIGMOD '21, page 2821–2828. Association for Computing Machinery, 2021. (Cited on 2)

[9] Marcelo Arenas and Jorge Pérez. Querying semantic web data with sparql. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 305–316, 2011. (Cited on 1)

[10] Pablo Barceló. Querying graph databases. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*, pages 175–188. ACM, 2013. (Cited on 2, 12)

[11] Pablo Barceló and Gaëlle Fontaine. On the data complexity of consistent query answering over graph databases. *Journal of Computer and System Sciences*, 88:164–194, 2017. (Cited on 2, 3, 10, 16, 22, 27, 28, 33)

[12] Pablo Barceló, Leonid Libkin, Anthony W Lin, and Peter T Wood. Expressive languages for path queries over graph-structured data. *ACM Transactions on Database Systems (TODS)*, 37(4):1–46, 2012. (Cited on 34)

[13] Pablo Barceló, Jorge Pérez, and Juan L Reutter. Relative expressiveness of nested regular expressions. *AMW*, 12:180–195, 2012. (Cited on 2, 12, 27)

[14] Paul Beame, Jerry Li, Sudeepa Roy, and Dan Suciu. Model counting of query expressions: Limitations of propositional methods. *arXiv preprint arXiv:1312.4125*, 2013. (Cited on 4)

[15] Leopoldo Bertossi. *Database Repairing and Consistent Query Answering*. Morgan & Claypool Publishers, 2011. (Cited on 24, 33)

[16] Meghyn Bienvenu. On the complexity of consistent query answering in the presence of simple ontologies. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012. (Cited on 33)

[17] Peter Buneman, Wenfei Fan, and Scott Weinstein. Path constraints in semistructured databases. *Journal of Computer and System Sciences*, 61(2):146–193, 2000. (Cited on 2)

[18] Diego Calvanese, Magdalena Ortiz, and Mantas Šimkus. Verification of evolving graph-structured data under expressive path constraints. In *19th International Conference on Database Theory (ICDT 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016. (Cited on 27)

[19] Jan Chomicki and Jerzy Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Information and Computation*, 197(1-2):90–121, 2005. (Cited on 10)

[20] L. da F. Costa, F. A. Rodrigues, G. Travieso, and P. R. Villas Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, jan 2007. (Cited on 2)

[21] Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, 2007. (Cited on 3)

[22] Nilesh Dalvi and Dan Suciu. Management of probabilistic data: foundations and challenges. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–12, 2007. (Cited on 4)

[23] Nilesh Dalvi and Dan Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *Journal of the ACM (JACM)*, 59(6):1–87, 2013. (Cited on 4)

[24] Wenfei Fan. Graph pattern matching revised for social network analysis. In *Proceedings of the 15th International Conference on Database Theory*, pages 8–21, 2012. (Cited on 1)

[25] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. A survey of community search over big graphs. *The VLDB Journal*, 29:353–392, 2020. (Cited on 2)

[26] Dieter Fensel, Umutcan Simsek, Kevin Angele, Elwin Huaman, Kaerle Elias, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. *Knowledge Graphs - Methodology, Tools and Selected Use Cases*. Springer, 2020. (Cited on 2)

[27] Tal Friedman and Guy Van den Broeck. Symbolic querying of vector spaces: Probabilistic databases meets relational embeddings. In *Conference on Uncertainty in Artificial Intelligence*, pages 1268–1277. PMLR, 2020. (Cited on 4)

[28] Michael R Garey and David S Johnson. Computers and intractability. *A Guide to the*, 1979. (Cited on 21)

[29] Amir Gilad, Aviram Imber, and Benny Kimelfeld. The consistency of probabilistic databases with independent cells, 2022. (Cited on 4)

[30] Eric Gribkoff, Guy Van den Broeck, and Dan Suciu. The most probable database problem. In *Proceedings of the First international workshop on Big Uncertain Data (BUDA)*, pages 1–7, 2014. (Cited on 4)

[31] Eric Gribkoff, Guy Van den Broeck, and Dan Suciu. Understanding the complexity of lifted inference and asymmetric weighted model counting. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014. (Cited on 4)

[32] Claudio Gutierrez and Juan F. Sequeda. Knowledge graphs. *Commun. ACM*, 64(3):96–104, 2021. (Cited on 2)

[33] Teresa W Haynes and Michael A Henning. Domination critical graphs with respect to relative complements. *Australasian Journal of Combinatorics*, 18:115–126, 1998. (Cited on 19)

[34] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. (Cited on 2)

[35] Benny Kimelfeld and Yehoshua Sagiv. Modeling and querying probabilistic XML data. *ACM SIGMOD Record*, 37(4):69–77, 2009. (Cited on 4)

[36] Xiang Lian, Lei Chen, and Shaoxu Song. Consistent query answers in inconsistent probabilistic databases. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 303–314, 2010. (Cited on 2, 4)

[37] Leonid Libkin, Wim Martens, and Domagoj Vrgoč. Querying graphs with data. *Journal of the ACM (JACM)*, 63(2):1–53, 2016. (Cited on 2, 12, 25)

[38] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks V.S. Lakshmanan, Wenjie Zhang, and Xuemin Lin. Efficient algorithms for densest subgraph discovery on large directed graphs. In *Proc. of ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 1051–1066. Association for Computing Machinery, 2020. (Cited on 2)

[39] Silviu Maniu, Reynold Cheng, and Pierre Senellart. An indexing framework for queries on probabilistic graphs. *ACM Transactions on Database Systems (TODS)*, 42(2):1–34, 2017. (Cited on 4)

[40] Dan Olteanu and Jiewen Huang. Using obdds for efficient query evaluation on probabilistic databases. In *International Conference on Scalable Uncertainty Management*, pages 326–340. Springer, 2008. (Cited on 3)

[41] Christos H Papadimitriou. Computational complexity. In *Encyclopedia of computer science*, pages 260–265. 2003. (Cited on 31)

[42] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. Holoclean: Holistic data repairs with probabilistic inference. *arXiv preprint arXiv:1702.00820*, 2017. (Cited on 2, 3, 7)

[43] Fabrizio Riguzzi, Elena Bellodi, Evelina Lamma, and Riccardo Zese. Reasoning with probabilistic ontologies. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015. (Cited on 4)

[44] Christopher De Sa, Ihab F. Ilyas, Benny Kimelfeld, Christopher Ré, and Theodoros Rekatsinas. A Formal Framework for Probabilistic Unclean Databases. In Pablo Barcelo and Marco Calautti, editors, *22nd International Conference on Database Theory (ICDT 2019)*, volume 127 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:18, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. (Cited on 2, 3, 7, 33)

[45] Anish Das Sarma, Omar Benjelloun, Alon Halevy, Shubha Nabar, and Jennifer Widom. Representing uncertain data: models, properties, and algorithms. *The VLDB Journal*, 18(5):989–1019, 2009. (Cited on 3)

[46] Asma Souihli and Pierre Senellart. Optimizing approximations of DNF query lineage in probabilistic XML. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 721–732. IEEE, 2013. (Cited on 4)

[47] Balder ten Cate, Gaëlle Fontaine, and Phokion G. Kolaitis. On the data complexity of consistent query answering. In *Proceedings of the 15th International Conference on Database Theory*, ICDT '12, pages 22–33, 2012. (Cited on 3, 28)

[48] Balder Ten Cate, Gaëlle Fontaine, and Phokion G Kolaitis. On the data complexity of consistent query answering. *Theory of Computing Systems*, 57(4):843–891, 2015. (Cited on 10)

[49] Guy Van den Broeck, Wannes Meert, and Adnan Darwiche. Skolemization for weighted first-order model counting. In *Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2014. (Cited on 4)

[50] Guy Van den Broeck, Dan Suciu, et al. Query processing on probabilistic data: A survey. *Foundations and Trends® in Databases*, 7(3-4):197–341, 2017. (Cited on 3)

[51] Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *Proc. ACM Symposium on Theory of Computing (STOC 82*, pages 137–146, 1982. (Cited on 24)

[52] Kai Zeng, Shi Gao, Barzan Mozafari, and Carlo Zaniolo. The analytical bootstrap: a new method for fast error estimation in approximate query processing. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 277–288, 2014. (Cited on 3)

9 Appendix

Proof of Theorem 20. This proof is analogous as the one provided for Theorem 18. The only difference is that we need to define $\mathcal{R}$ so that it actually removes the assignment of the data-graph by adding all possible edges of the form $(\star, \text{VALUE}, x_i)$ for $\star \in \{\bot, \top\}$ and $1 \leq i \leq n$. Such $\mathcal{R}$ can be defined as

$$\mathcal{R}(G)(H) = 2^{-(|\Sigma_e||V_G|^2 - |E_G|)}$$

if $V_G = V_H$ and $G \subseteq H$. Otherwise, $\mathcal{R}(G)(H) = 0$.

Now, for the reduction, given a 3CNF formula ϕ we map it to the data-graph $G_{\phi,f}$ where f intuitively "assigns" both boolean values to each variable. Deciding if there is some data-graph such that $\mathcal{B}_{\mathcal{I},\mathcal{R}} > \frac{1}{2}^n$ is equivalent to deciding if there exists a satisfying assignment for ϕ. □

Proof of Theorem 22. Even though the main ideas are the same as in Theorem 18, for this proof we have to define a somewhat different $\mathcal{R}$ and reduction. First, $\mathcal{R}$ will once again delete the assignment represented by the data-graph, but in this case this information needs to be represented through the edge labels. That is, we consider that a data-graph represents a 3CNF formula ϕ alongside an assignment f of its variables if we have the m clause and n variable nodes with the corresponding IS_LITERAL and IS_LITERAL_NEGATED, the two boolean nodes $\bot$ and $\top$ with an edge between them, but now the assignment f will be present in the data-graph in the following sense: each variable node will have an edge from itself to $\bot$ and $\top$, but only one of those will have the label CHOSEN, while the other one will have the label UNCHOSEN. Then, we can represent the "deletion" of an assignment satisfying ϕ by turning all CHOSEN edges to UNCHOSEN ones. Observe that the probabilistic database $\mathcal{I}$ can be defined in an analogous way. Finally, the realization model $\mathcal{R}$ is defined as:

$$\mathcal{R}(G)(H) = 2^{-|E_G^{\neg\text{UNCHOSEN}}|}$$

if $V_G = V_H$, G has the same edges as H up to a renaming of the edge labels and the only edges with different labels have a label UNCHOSEN in H. In this formula, $E_G^{\neg\text{UNCHOSEN}}$ denotes the set of edges from G with label different from UNCHOSEN.

As in the proof of Theorem 18, we complete the reduction from 3SAT: given a $3CNF$ formula ϕ we build the data-graph $G' = G_{\phi,f}$, where f assigns to each variable x_i both boolean values. Moreover, we consider $b = \frac{1}{2^n}$. □

Proof of Lemma 35. Let η be a Reg-GXPath formula, $G = (V_G, L_G, D_G)$ a data-graph of n nodes and A a set of data values such that $|A| = n$ and $\Sigma_n^\eta \cap A = \emptyset$. Let $C = \{\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_k\}$ be the set of all data values present in G that are not in Σ_n^η, and let $G' = (V_G, L_G, D_{G'})$ be a new data-graph such that

$$D_{G'}(v) = \begin{cases} \mathrm{D}_i & D_G(v) = \mathrm{C}_i \text{ for some } i \\ D_G(v) & \textit{otherwise} \end{cases}$$

We prove by induction on the formula's structure that:

1. If η is a path expression, then $(v, w) \in [\![\eta]\!]_G \iff (v, w) \in [\![\eta]\!]_{G'}$.
2. If η is a node expression, then $v \in [\![\eta]\!]_G \iff v \in [\![\eta]\!]_{G'}$

The only interesting cases are those in which the expression interacts with data values, since the edges of the data-graph were not modified. Observe that the theorem is true for the base cases: if $\eta = [\mathrm{C}^=]$, then clearly $[\![\eta]\!]_G = [\![\eta]\!]_{G'}$, since those nodes with data values in Σ_n^η kept their original data. Now, if $\eta = [\mathrm{C}^\neq]$ the theorem also holds: if $v \in [\![\mathrm{C}^\neq]\!]_G$ then $D_G(v)$ is in $\Sigma_n^\eta \setminus \{\mathrm{C}\}$ and was not modified in G', or rather it is not in Σ_n^η, and belongs to A. Since $A \cap \Sigma_n^\eta = \emptyset$ we conclude that $v \in [\![\mathrm{C}^\neq]\!]_{G'}$.

For the inductive cases, we consider:

- $\eta \equiv \langle \alpha = \beta \rangle$: this case follows easily by noticing that, if two nodes had the same data value in G then they also have the same data value in G'.
- $\eta \equiv \langle \alpha \neq \beta \rangle$: this case follows by observing that, if two nodes had a different data value in G, then they also contain a different data value in G'.

□

Proof of Lemma 37. Let $\varphi \in \text{Reg-GXPath}_{core}^{pos}$ be a node expression that does not use the $\langle \alpha = \beta \rangle$ subexpression, $G = (V_G, L_G, D_G)$ a data-graph with an origin o and $\delta : \Sigma_n \times \Sigma_n \to \mathbb{N}_0$ a function such that for each $c \in \Sigma_n$ it is possible to efficiently enumerate $\mathrm{D}_1, \mathrm{D}_2, \ldots$ where $\delta(\mathrm{D}_i, c) \leq \delta(\mathrm{D}_{i+1}, c)$ and every data value D belongs to the enumeration.

We want to show that, if there exists a data-graph $H = (V_G, L_G, D_H)$ such that $H, o \models \varphi$ and $\sum_{v \in V_G} \delta(D_H(v), D_G(v))$ is minimized across all data-graphs equal to G up to its data values that also satisfy φ at the origin, then there is another data-graph $H' = (V_G, L_G, D_{H'})$ such that $H', o \models \varphi$, H' also minimizes $\sum_{v \in V_G} \delta(D_{H'}(v), D_G(v))$ and for each data value C in G that was changed into D in H' where D is not mentioned by φ it is satisfied that $\mathrm{D} = \mathrm{D}_j$ for $j \leq n + |\Sigma_n^\varphi|$ where D_i is the ith data value in the enumeration $\mathrm{D}_1, \mathrm{D}_2, \ldots$ such that $\delta(\mathrm{D}_i, \mathrm{C}) \leq \delta(\mathrm{D}_{i+1}, \mathrm{C})$.

To prove this, it suffices to find, given H and a node v such that $D_H(v) \notin \Sigma_n^\varphi$ and $D_H(v) \neq \mathrm{D}_j$ for all $j \leq n + |\Sigma_n^\varphi|$ where the D_j are the numeration of δ for $D_G(v)$, a new data value D such that $\mathrm{D} = \mathrm{D}_j$ for $j \leq n + |\Sigma_n^\varphi|$, the data-graph obtained by considering H and modifying the data value of v to D also satisfies φ when evaluated at the origin, and the summation involving δ is still minimized.

Let Σ_n^H be the set of data values that are contained by some node of H. Observe that the set $A = \{\mathrm{D}_j : 1 \leq j \leq n + |\Sigma_n^\varphi|, \mathrm{D}_j \notin \Sigma_n^\varphi\}$ has size at least n. Then, since $D_H(v) \notin A$ there is at least one data value $\mathrm{D} \in A \setminus \Sigma_n^H$. We claim that the data-graph $H' = (V_H, L_H, D_{H'})$ where $D_{H'}(w) = D_H(w)$ unless $w = v$, in which case $D_{H'}(v) = \mathrm{D}$, satisfies all the desired conditions.

Clearly, δ is also minimized in H', since we only chose a transition with smaller cost. Using the following claim we conclude that H' satisfies φ at the origin. □

Claim: Let G be a data-graph and η an expression from Reg-GXPath^{pos} without data comparison with equality (i.e. the subexpression $\langle \alpha = \beta \rangle$). Then, given a node v such that $D_G(v) \notin \Sigma_n^{\varphi}$, we can define the data-graph $G' = (V_G, L_G, D_{G'})$ where $D_{G'}(w) = D_G(w)$ for $w \neq v$, and $D_{G'}(v) = \mathrm{D} \in \Sigma_n \setminus (\Sigma_n^{\varphi} \cup \Sigma_n^G)$. Then, $[\![\eta]\!]_G = [\![\eta]\!]_{G'}$.

Proof of the Claim: We prove this by structural induction. As before, the only interesting cases are those in which the subexpression interacts with data values. For the base cases:

- $\eta \equiv \mathrm{C}^{=}$: if $w \in [\![\mathrm{C}^{=}]\!]_G$ we can conclude that $w \in \Sigma_n^{\varphi}$, which implies that $w \neq v$. Then, $D_{G'}(w) = D_G(w)$, and $w \in [\![\mathrm{C}^{=}]\!]_{G'}$. The other direction can be seen in the same way.
- $\eta \equiv \mathrm{C}^{\neq}$: if $w \in [\![\mathrm{C}^{\neq}]\!]$, then $D_G(w) \notin \Sigma_n^{\varphi}$, and therefore $D_{G'}(w) \notin \Sigma_n^{\varphi}$ and $w \in [\![\mathrm{C}^{\neq}]\!]_{G'}$. The other direction follows with similar arguments.

For the recursive cases, we only need to consider $\eta \equiv \langle \alpha_1 \neq \alpha_2 \rangle$, since the other ones are not affected because the topology of the graph remained untouched. Observe that if $w \in [\![\langle \alpha_1 \neq \alpha_2 \rangle]\!]_G$, then there exists nodes z_1, z_2 such that $(w, z_i) \in [\![\alpha_i]\!]_G$ and $D(z_1)_G \neq D(z_2)_G$. We know that $(w, z_i) \in [\![\alpha_i]\!]_{G'}$ by induction, so we can conclude by proving that $D_{G'}(z_1) \neq D_{G'}(z_2)$. Observe that this inequality could only have been altered if $z_i = v$ for some i. But, in that case, the inequality still holds, since $D_{G'}(v) \notin \Sigma_n^H$ by construction. The other direction can be proven in the exact same way. □