ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-1414

Complexity Results for Probabilistic Datalog$^{\pm}$

İsmail İlkan Ceylan[1] **Thomas Lukasiewicz**[2] **Rafael Peñaloza**[3]

Abstract. We study the query evaluation problem in probabilistic databases in the presence of probabilistic existential rules. Our focus is on the Datalog$^{\pm}$ family of languages for which we define the probabilistic counterpart using a flexible and compact encoding of probabilities. This formalism can be viewed as a generalization of probabilistic databases, as it allows to generate new facts from the given ones, using so-called tuple-generating dependencies, or existential rules. We study the computational cost of this additional expressiveness under two different semantics. First, we use a conventional approach and assume that the probabilistic knowledge base is consistent and employ the standard possible world semantics. Thereafter, we introduce a probabilistic inconsistency-tolerant semantics, which we call inconsistency-tolerant possible world semantics. For both of these cases, we provide a thorough complexity analysis relative to different languages, drawing a complete picture of the complexity of probabilistic query answering in this family.

1 INTRODUCTION

Recent years have lead to a significant increase in the number of application domains that generate large volumes of *uncertain data*. This has paved the way for a number of *systems* tailored towards such domains; most notably for large knowledge bases: Yago [22], Nell [31], DeepDive [36], Google's Knowledge Vault [17], and Microsoft's Probase [42] are systems containing a large amount of uncertain data. These systems are substantially based on the foundations of *probabilistic databases (PDBs)* [37]. Arguably, PDBs provide the state-of-the-art means for *modeling*, *storing*, and *processing* data in the presence of uncertainty.

Enriching databases with *ontological knowledge* is a common paradigm [33], as it allows one to deduce facts that are not explicitly specified in the database. The most widely studied languages for achieving such sophisticated data access are based on *description logics* (DLs) [2] and *existential rules* [9, 8]. Following this tradition, we study *probabilistic query entailment* under existential rules (tuple-generating dependencies) relative to a database. We focus on a particular family of existential rule languages, which is also referred to as Datalog$^{\pm}$ [9, 8].

Our framework is rather general: We assume a set of probabilistic events and annotate the facts and the rules with a Boolean expression formed over these events, which we call *contexts*. This context-based abstraction allows a compact specification of a probability distribution over the knowledge base. Similar approaches have been used in *knowledge representation* [32] and are also related to *data provenance* and *lineage* [23, 32, 37] in PDBs.

The most common semantics for PDBs is the *possible world semantics*: a PDB factorizes into a set of possible worlds, i.e., classical databases, each of which is then associated with a probability. This semantics is also used in probabilistic logic programming (see, e.g., ProbLog [16]) and is closely related to Poole's independent choice logic [34]. We first study probabilistic query entailment in Datalog$^{\pm}$ under this semantics with a conventional assumption, i.e., the assumption that the probabilistic knowledge base is *consistent*.

Datalog$^{\pm}$ programs can clearly lead to inconsistencies, as negative constraints, such as $\forall x\, \mathtt{P}(x) \wedge \mathtt{R}(x) \rightarrow \bot$, are part of these programs. The obvious question is, of course, whether forcing the consistency assumption is always feasible? We answer this question negatively: PDBs are typically constructed in an automated manner; therewith, it is not easy to control which tuple is to be added to the database next. Suppose, e.g., that both atoms $\mathtt{P}(u)$ and $\mathtt{R}(u)$ are obtained with a positive probability. Clearly, adding both atoms would lead to an inconsistency, as the disjointness imposed by the rule will then be invalidated; i.e., we either throw away one of these atoms, or the whole knowledge base becomes inconsistent.

One way of tackling this problem is to simply ignore the inconsistent worlds imposed by the knowledge base, and as such, to slightly change the possible world semantics to only consider consistent worlds. We argue that this is not a solution to the problem, but rather a patch, and show that considering only consistent worlds could lead to loss of valuable information. In other words, inconsistent worlds may produce meaningful answers that are lost, as they can not be captured with an adequate semantics. Thus, to retrieve as much valuable information as possible, we base ourselves on the foundations of *inconsistency-tolerant* reasoning, which is well-understood both in the context of DLs [26, 5, 6] and Datalog$^{\pm}$ [28, 29, 27]. A well-known approach in inconsistency-tolerant reasoning is based on *repairing* the knowledge base by minimally removing some facts. As there can be many different minimal repairs (see the example above), the safe consequences are considered to be those that follow from *every* possible repair. In this paper, we adopt the *generalized repair (GR)* semantics from a recent work [18], which allows repairs both on the database and on the program. Based on the GR semantics, we define the *inconsistency tolerant possible world semantics*.

For both semantic approaches, we provide a through complexity analysis relative to different existential rule languages, drawing a complete picture of the complexity of probabilistic query entailment in Datalog$^{\pm}$. The most central class for our complexity analysis is the class PP [20], which we describe in detail. Briefly stated, our results show an analogous behavior to the classical case, i.e., moving to inconsistency-tolerant semantics can put the complexity of reasoning one level higher in the respective hierarchy.

[1] TU Dresden, Germany, email: ceylan@tcs.inf.tu-dresden.de
[2] University of Oxford, UK, email: thomas.lukasiewicz@cs.ox.ac.uk
[3] Free University Bozen-Bolzano, Italy, email: rafael.penaloza@unibz.it

2 MOTIVATION AND BACKGROUND

We enrich databases with ontological knowledge allowing to access probabilistic data over a logical abstraction. We concentrate on existential rule languages, also known as tuple-generating dependencies.

2.1 Existential Rules and Datalog$^{\pm}$

We recall some basics on existential rules from the context of Datalog$^{\pm}$ [9, 8] and briefly introduce conjunctive query answering under existential rules.

General. Consider (possibly infinite) mutually disjoint sets $\mathbf{R}$ of *predicates*, $\mathbf{C}$ of *constants*, $\mathbf{V}$ of *variables*, and $\mathbf{N}$ of *nulls*. A *term* t is a constant, a null, or a variable. An *atom* is an expression of the form $\mathtt{P}(t_1, \ldots, t_n)$, where $\mathtt{P}$ is an n-ary predicate, and $t_1, \ldots, t_n$ are terms. A *variable-free atom* does not contain any variables as terms, and a *ground atom* is an atom that contains only constants as terms. An *instance* I is a (possibly infinite) set of variable-free atoms. A *database* $\mathcal{D}$ is a finite set of ground atoms.

Programs. A *tuple-generating dependency (TGD)* (or *existential rule*) σ is a first-order formula $\forall \mathbf{x}\, \varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \mathtt{P}(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} \cup \mathbf{y} \subset \mathbf{V}$, $\varphi(\mathbf{x})$ is a conjunction of atoms, and $\mathtt{P}(\mathbf{x}, \mathbf{y})$ is an atom; $\varphi(\mathbf{x})$ is the *body* of σ, denoted $body(\sigma)$, while $\mathtt{P}(\mathbf{x}, \mathbf{y})$ is the *head* of σ, denoted $head(\sigma)$.[4]

A *negative constraint (NC)* ν is a first-order formula of the form $\forall \mathbf{x}\, \varphi(\mathbf{x}) \rightarrow \bot$, where $\mathbf{x} \subset \mathbf{V}$, $\varphi(\mathbf{x})$ is a conjunction of atoms, called the *body* of ν, denoted $body(\nu)$, and $\bot$ denotes the truth constant *false*; i.e., a contradiction. A Datalog$^{\pm}$ *program* is a finite set Σ of TGDs and NCs. For brevity, we often omit the universal quantifiers in front of TGDs and NCs, and write simply, e.g., $\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \mathtt{P}(\mathbf{x}, \mathbf{y})$. Moreover, we often speak simply of *programs* when referring to Datalog$^{\pm}$ programs.

Semantics. The semantics of programs is defined via homomorphisms. Briefly, a *homomorphism* is a substitution $h\colon \mathbf{C} \cup \mathbf{N} \cup \mathbf{V} \rightarrow \mathbf{C} \cup \mathbf{N} \cup \mathbf{V}$ that behaves as the identity over $\mathbf{C}$. For a homomorphism h and a set of variables $\mathbf{x}$, we denote by $h_{|\mathbf{x}}$ the restriction of h to $\mathbf{x}$. The instance I *satisfies* the TGD σ, written $I \models \sigma$, if for every homomorphism h such that $h(\varphi(\mathbf{x})) \subseteq I$, there exists $h' \supseteq h_{|\mathbf{x}}$ such that $h'(\mathtt{P}(\mathbf{x}, \mathbf{y})) \in I$. The instance I *satisfies* the NC ν, written $I \models \nu$, if there is no homomorphism h such that $h(\varphi(\mathbf{x})) \subseteq I$. Given a program Σ, I *satisfies* Σ, written $I \models \Sigma$, if I satisfies each TGD and NC of Σ. I is a *model* of the program Σ relative to the database $\mathcal{D}$, if $\mathcal{D} \subseteq I$ and $I \models \Sigma$. We denote the set of all models of Σ relative to $\mathcal{D}$ as $mods(\mathcal{D}, \Sigma)$.

Unions of Conjunctive Queries. A *conjunctive query (CQ)* is an existentially quantified formula $\exists \mathbf{x}\, \psi(\mathbf{x})$, where ψ is a conjunction of atoms. Consider, e.g., the query

$$\mathtt{q}_1(x) = \exists y\, \mathtt{StarredIn}(x, y) \wedge \mathtt{Mov}(y)\,,$$

which asks for individuals that starred in a movie. Notice that x is a free variable in $\mathtt{q}_1$, also called an *answer variable*. A *Boolean conjunctive query (BQ)* is a CQ without any free variables. An example is the query

$$\mathtt{q}_2 = \exists x, y\, \mathtt{StarredIn}(x, y) \wedge \mathtt{Mov}(y)\,,$$

which asks whether there exists an individual that starred in a movie.

[4] Notice that our definition of TGDs requires the head to contain only one atom. This restriction is made w.l.o.g., as a TGD with a conjunction of atoms in the head can be equivalently represented by a set of single-atom-headed TGDs [8].

A *union of Boolean conjunctive queries (UCQ)* $\mathtt{Q}$ is a disjunction of BQs. For notational convenience, we write $\mathtt{Q}$ to represent UCQs and $\mathtt{q}$ to represent BQs. If we consider queries with free variables, we make this explicit and write $\mathtt{Q}(\mathbf{x})$, or $\mathtt{q}(\mathbf{x})$, respectively.

Query Semantics. The answers to a CQ $\mathtt{q}(\mathbf{x})$ over an instance I, denoted $\mathtt{q}(I)$, is the set of all mappings Θ from $\mathbf{x}$ to the constants in I such that $\mathtt{q}(\Theta(\mathbf{x})) \in I$. A Boolean query $\mathtt{q}$ has a positive answer over I, denoted $I \models \mathtt{q}$, if $\mathtt{q}(I) \neq \emptyset$. Given a database $\mathcal{D}$ and a program Σ, the answers we consider are those that are true in *all* models of Σ relative to $\mathcal{D}$. Formally, the *answer* to a CQ $\mathtt{q}$ w.r.t. $\mathcal{D}$ and Σ is the set of tuples $ans(\mathtt{q}, \mathcal{D}, \Sigma) = \bigcap_{I \in mods(\mathcal{D}, \Sigma)} \{\mathbf{t} \mid \mathbf{t} \in \mathtt{q}(I)\}$. The answer to a BQ $\mathtt{q}$ is *positive*, denoted $\mathcal{D} \cup \Sigma \models \mathtt{q}$, if $ans(\mathtt{q}, D, \Sigma) \neq \emptyset$. These notions are generalized to the class of UCQs in the obvious way. Consider a database

$$\mathcal{D} = \{\mathtt{Actor}(\mathtt{alPacino}), \mathtt{StarredIn}(\mathtt{pMiller}, \mathtt{cw}), \mathtt{Mov}(\mathtt{cw})\},$$

which asserts that *Al Pacino* is an actor, and that *Penelope Miller* has starred in a movie. The query $\mathtt{q}_1(x)$ produces only `pMiller` as an answer on $\mathcal{D}$. In the presence of the program

$$\Sigma = \{\langle\, \mathtt{Actor}(x) \rightarrow \exists y\, \mathtt{StarredIn}(x, y), \mathtt{Mov}(y)\rangle\},$$

a new tuple (`alPacino`) is generated, and thus both `alPacino` and `pMiller` become answers to $\mathtt{q}_1(x)$.

2.2 Computational Properties of Existential Rules

In general, it is undecidable whether a BQ has an answer or not w.r.t. a database $\mathcal{D}$ and a program Σ [4]. To regain decidability, many different restrictions on the class of allowed TGDs have been proposed. The most important (syntactic) restrictions studied in the literature are guardedness [8], stickiness [9], and acyclicity, along with their "weak" counterparts, namely weak guardedness [8], weak stickiness [9], and weak acyclicity [19], respectively.

A TGD σ is *guarded*, if there exists an atom $\mathbf{a} \in body(\sigma)$ that contains (or "guards") all the body variables of σ. The class of guarded TGDs, denoted G, is defined as the family of all possible sets of guarded TGDs. A key subclass of guarded TGDs are the so-called linear TGDs with just one body atom, which is automatically the guard. The class of linear TGDs is denoted by L. *Weakly guarded* TGDs extend guarded TGDs by requiring only the body variables that are considered "harmful" to appear in the guard (see [8] for full details). The associated class of TGDs is denoted WG. It is easy to verify that $\mathsf{L} \subset \mathsf{G} \subset \mathsf{WG}$, in terms of the sets of TGDs they contain.

Stickiness is inherently different from guardedness, and its central property can be described as follows: variables that appear more than once in a body (i.e., join variables) must always be propagated (or "stuck") to the inferred atoms. A TGDs that enjoys this property is called *sticky*, and the class of sticky TGDs is denoted by S. Weak stickiness generalizes stickiness by considering only "harmful" variables, and defines the class WS of *weakly sticky* TGDs. Observe that $\mathsf{S} \subset \mathsf{WS}$.

A set Σ of TGDs is *acyclic* (and belongs to the class A), if its predicate graph is acyclic. Equivalently, an acyclic set of TGDs can be seen as a non-recursive set of TGDs. We say Σ is *weakly-acyclic*, if its dependency graph enjoys a certain acyclicity condition, which guarantees the existence of a finite canonical model; the associated class is denoted WA. Clearly, $\mathsf{A} \subset \mathsf{WA}$. Interestingly, it also holds that $\mathsf{WA} \subset \mathsf{WS}$ [9].

Another key fragment of TGDs, which deserves our attention, are the so-called *full* TGDs, i.e., TGDs without existentially quantified

variables. Their corresponding class is denoted as F. Restricting full TGDs to satisfy linearity, guardedness, stickiness, or acyclicity yields the classes L, G, S, and A, respectively. A known relation between these classes is that F $\subset$ WA [19] and F $\subset$ WG [8]. We extend all these notions to programs Σ in the obvious way: by considering the properties satisfied by the TGDs in Σ. Thus, for instance, Σ is guarded, if all the TGDs in Σ are guarded.

When analysing the complexity of query answering, we consider all these classes of programs unless explicitly mentioned otherwise. To obtain a fine-grained analysis of the computational complexity, we follow Vardi's taxonomy [40], as described next. The *combined complexity* of UCQ answering is calculated by considering all the components, i.e., the database, the program, and the query, as part of the input. The *bounded-arity combined complexity* (or simply *ba-combined complexity*) assumes that the arity of the underlying schema (i.e., the maximum arity of the predicates in $\mathbf{R}$) is bounded by an integer constant. In the context of description logics (DLs), the combined complexity in fact refers to the *ba*-combined complexity, since, by definition, the arity of the underlying schema is at most two. The *fixed-program combined complexity* (or simply *fp-combined complexity*) is calculated by considering the program (i.e., the set of TGDs and NCs) as fixed, while the *data complexity* additionally assumes that the query is fixed.

Table 1 summarizes the known complexity results for query entailment in the different classes of programs that we consider. These results will provide the basis for analysing the complexity of probabilistic Datalog$^{\pm}$ programs in the following sections.

	Data	**Comb.**	***ba*-comb.**	***fp*-comb.**
L, LF, AF	in AC^0	PSPACE	NP	NP
G	P	2EXP	EXP	NP
WG	EXP	2EXP	EXP	EXP
S, SF	in AC^0	EXP	NP	NP
F, GF	P	EXP	NP	NP
A	in AC^0	NEXP	NEXP	NP
WS, WA	P	2EXP	2EXP	NP

Table 1: Complexity of BQ answering [27]. All entries except for "in AC^0" are completeness ones, where hardness in all entries but the *fp*-combined ones holds even for ground atomic BQs.

2.3 Complexity of Standard Probabilistic Inference

Our approach is based on annotating the facts in the database and the rules in the Datalog$^{\pm}$ program with Boolean events, which we call *contexts*. Here, we briefly introduce the basic notions, our assumptions, and the complexity of probabilistic Boolean inferences.

Consider a finite set of elementary events $\mathbf{E} = \{e_1, ..., e_n\}$. A *world* is a conjunction $w = s_1 \wedge ... \wedge s_n$ where $s_i, 1 \leq i \leq n$, is either the event e_i or its negation $\neg e_i$. A *context* is a Boolean combination of elementary events, i.e., if κ_1 and κ_2 are contexts, then so is $\neg\kappa_1$ and $\kappa_1 \wedge \kappa_2$.

Contexts encompass the probabilistic component of our formalism. For representing the probability distribution of events and contexts, we do not restrict to any specific probabilistic model, but rather consider any representation for which deciding whether $\mathrm{P}(\kappa) > p$ for some value $p \in [0, 1)$ is PP-complete. Further details on the complexity class PP and its relation to other complexity classes can be found in Section 3.1.

3 PROBABILISTIC DATALOG$^{\pm}$

To define our probabilistic extension of Datalog$^{\pm}$, we annotate all the rules and negative constraints with contexts, which will be interpreted through the probability distribution. Similarly, all the atoms in a probabilistic database are associated with a context as well.

Definition 1 (Probabilistic Datalog$^{\pm}$) A *probabilistic TGD* is an expression of the form $\langle \sigma : \kappa \rangle$, where σ is a TGD, and κ is a context. Analogously, a *probabilistic negative contraint* is of the form $\langle \nu : \kappa \rangle$, where ν is a negative constraint, and κ is a context. A *probabilistic program* Γ is a finite set of probabilistic TGDs and probabilistic negative contraints.

A *probabilistic* atom is of the form $\langle \ell : \kappa \rangle$, where ℓ is an atom, and κ is a context. A *probabilistic database* $\mathcal{P}$ is a finite set of probabilistic atoms. A *probabilistic knowledge base* is a pair $\mathcal{K} = (\Gamma, \mathcal{P})$ that represents a probabilistic program Γ relative to a probabilistic database $\mathcal{P}$.

We extend the special cases of Datalog$^{\pm}$ programs defined in the previous section to probabilistic programs in the obvious way. That is, the probabilistic program Γ is *guarded* if the Datalog$^{\pm}$ program $\{\lambda \mid \langle \lambda : \kappa \rangle \in \Gamma\}$ is guarded, and analogously for linear, sticky, acyclic, and full programs, and their weak versions. For a class $\mathcal{L}$ of Datalog$^{\pm}$ programs, we denote by $\Upsilon_{\mathcal{L}}$ its associated class of probabilistic programs. Thus, for instance Υ_{G} is the class of all guarded probabilistic programs. Consider the probabilistic program Γ_m relative to the PDB $\mathcal{P}_m$ given in Figure 1. It asserts that actors star in at least one movie and that actors and movies are disjoint entities. Both expressions hold in the *global context* $\top$. To ease reading, we usually omit the global context from the expressions.

Intuitively, a probabilistic program relative to a PDB compactly encodes a finite number of classical programs relative a classical database, each of which associated with a different context, and therefore a number of worlds. This semantics is commonly referred to as the *possible world semantics*.

Definition 2 (Possible Worlds) Let $\mathcal{K} = (\Gamma, \mathcal{P})$ be a probabilistic knowledge base. Every world w induces a classical knowledge base $\mathcal{K}_{|w} = (\Gamma_{|w}, \mathcal{P}_{|w})$ where

$$\Gamma_{|w} = \{\lambda \mid \langle \lambda : \kappa \rangle \in \Gamma, w \models \kappa\},$$
$$\mathcal{P}_{|w} = \{\ell \mid \langle \ell : \kappa \rangle \in \mathcal{P}, w \models \kappa\}.$$

A probabilisitic knowledge base $\mathcal{K}$ is *consistent*, if all the worlds induced by $\mathcal{K}$ (with positive probability) are consistent.

The probabilistic program Γ_m relative to $\mathcal{P}_m$ encodes exponentialy many worlds on the size of the context variables. For instance, given the world w_1 (see Figure 1), $\mathcal{P}_{|w_1}$ contains all tuples from `Actors` and `Movies`, but none from `StarredIn`. Similarly, as the rules in Γ_m are global (i.e., they hold in every world), $\Gamma_{|w_1}$ contains both rules.

Observe also that *tuple-independent* probability models are a special case of our abstraction, where every annotation is independent from others. In this case, one can directly write probability values, instead of the contexts with their independent probabilities.

Definition 3 (Query Semantics) Let $\mathcal{K} = (\Gamma, \mathcal{P})$ be a probabilistic knowledge base, the *probability of a UCQ* Q is given by:

$$\mathrm{P}_{\mathcal{K}}(\mathsf{Q}) = \sum_{\mathcal{K}_{|w} \models \mathsf{Q}} \mathrm{P}(w),$$

Given a query Q and $p \in (0, 1]$, *probabilistic query entailment* is the problem of deciding whether $\mathrm{P}_{\mathcal{K}}(\mathsf{Q}) \geq p$.

Actor	Pr
alPacino	a_1
rDeNiro	a_2
mPfeiffer	a_3

Movies	Pr
carlitosWay	m_1
godfather	m_2
taxiDriver	m_3

StarredIn		Pr
alPacino	carlitosWay	$\neg s_1 \wedge s_2$
alPacino	godfather	$s_3 \wedge \neg s_4$
rDeNiro	godfather	$\neg s_5 \wedge s_6$
pMiller	carlitosWay	$\neg s_7 \wedge s_1$

Γ_m

$R_1 : \langle \texttt{Actor}(x) \rightarrow \exists y\, \texttt{StarIn}(x, y), \texttt{Mov}(y) \rangle$

$R_2 : \langle \texttt{Actor}(x), \texttt{Mov}(x) \rightarrow \bot \rangle$

#	Worlds	Pr
w_1	$\{ a_1, a_2, a_3, m_1, m_2, m_3, s_1, ..., s_7 \}$	.73
w_2	$\{ \neg a_1, a_2, a_3, m_1, m_2, m_3, s_1, ..., s_7 \}$	.11
w_3	...	.12
w_4	...	.25
...	...	...
w_n	$\{ \neg a_1, \neg a_2 , \neg a_3, \neg m_1, \neg m_2, \neg m_3, \neg s_1, ..., \neg s_7 \}$	.01

Figure 1: The probabilistic database $\mathcal{P}_m$ (depicted using tables) and the probabilistic program $\Gamma_m = \{R_1, R_2\}$ composed of a TGD (R_1) and an NC (R_2). The contexts are defined over the elementary events $\mathbf{E} = \{a_1, a_2, a_3, m_1, m_2, m_3, s_1, \ldots, s_7\}$.

Briefly, a UCQ describes a desired pattern for a given knowledge base, and query entailment is then the task of deciding whether the specified pattern holds in this KB. Probabilistic query entailment factorizes this decision over different KBs, and we are interested in learning how likely it is for a UCQ to be entailed. Consider, for instance, the probabilistic KB $\mathcal{K}_m = (\Gamma_m, \mathcal{P}'_m)$, where

$$\mathcal{P}'_m = \{ \langle \texttt{Actor}(\texttt{alPacino}, \texttt{godfather} : 0.5) \rangle , \\ \langle \texttt{Actor}(\texttt{rDeNiro}, \texttt{godfather}) : 0.5 \rangle \},$$

and Γ_m given as before. The query q_2 would return the probability 0.75 on the program Γ_m relative to the PDB $\mathcal{P}'_m$. Notice that the only world that does not satisfy the query is $\{\texttt{Movie}(\texttt{gf})\} \cup \Gamma_m$, and this world has the probability 0.25. It is easy to see that q_2 would evaluate to 0, if it is posed only on $\mathcal{P}'_m$.

3.1 Complexity Classes and Assumptions

For the sake of readability, we briefly recall some of the non-standard complexity classes that we consider, and their relation to other classical complexity classes. The most typical counting complexity class

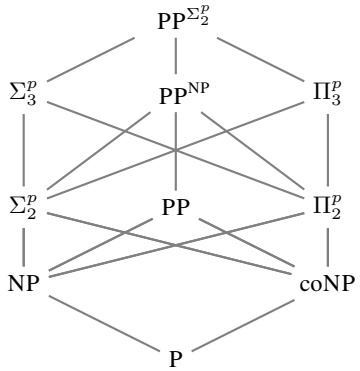

Figure 2: A portion of the counting polynomial-time hierarchy.

is #P [39], which is a functional complexity class originally introduced in the context of counting problems. The corresponding decision class PP [20] defines the set of languages recognized by a polynomially bounded non-deterministic Turing machine (TM) that accepts an input if and only if more than half of the computation paths are accepting [38]; such machines are usually called PP TMs. We also consider PP^{NP} (resp., $\text{PP}^{\Sigma_2^p}$, PP^{NEXP}), which as usual corresponds to languages that can be recognized by a PP TM, with an NP (resp., Σ_2^p, NEXP) oracle. Most of these classes belong to the counting polynomial-time hierarchy [41], which is partially illustrated in Figure 2 along with the first levels of the polynomial hierarchy. The following relations between complexity classes are a consequence of the relationships depicted in Figure 2 and will also be useful throughout the rest of this paper:

$$\text{PP}^{\Sigma_2^p} \subseteq \text{PSPACE} \subseteq \text{EXP} \subseteq \text{NEXP} \subseteq \text{P}^{\text{NEXP}} \subseteq \text{PP}^{\text{NEXP}} \subseteq \text{2EXP}$$

3.2 Complexity Results

We will consider the complexity of query answering w.r.t. the different classes of probabilistic programs relative to different languages.

	Data	**Comb.**	*ba*-**comb.**	*fp*-**comb.**
L, LF, AF	PP	PSPACE	PP^{NP}	PP^{NP}
G	PP	2EXP	EXP	PP^{NP}
WG	EXP	2EXP	EXP	EXP
S, SF	PP	EXP	PP^{NP}	PP^{NP}
F, GF	PP	EXP	PP^{NP}	PP^{NP}
A	PP	NEXP	NEXP	PP^{NP}
WS, WA	PP	2EXP	2EXP	PP^{NP}

Table 2: Complexity of probabilistic entailment

We start with a general result that provides some bounds for the complexity of query entailment in probabilistic KBs parameterized on the complexity of its classical counterpart.

Theorem 4 *Let $\mathcal{L}$ be a class of Datalog$^{\pm}$ programs, and* k *be the complexity of query entailment in $\mathcal{L}$ relative to databases. Then, probabilistic query entailment in $\Upsilon_{\mathcal{L}}$ relative to probabilistic databases is (i)* k*-hard, (ii)* PP*-hard, and (iii) in* PP^{k}.

PP-hardness follows from the hardness of standard probabilistic inference. The full proof shows a construction of a probabilistic KB upon which standard query entailment can be decided, which proves k-hardness. [5] Membership to PP^{k} follows mainly from the observation that a probabilistic knowledge base is a factorized representation of exponentially many classical knowledge bases. Thus, it is possible to solve the problem (after properly adjusting the probabilities of the worlds) by deciding whether the majority of the oracle calls that decide classical query entailment return true.

We analyze the consequences of Theorem 4. Observe first that if k is a deterministic class that contains PP, then $\text{PP}^{\text{k}} = \text{k}$ and thus Theorem 4 directly provides tight complexity bounds. Notice that this is the case w.r.t. the combined complexities for all the classes except A. In the case of the class A, the complexity of query entailment

[5] For ease of presentation, we excluded the proofs from the main text; for the interested reader, we refer to the appendix of this paper.

is complete w.r.t. to the class NEXP, and it is not known whether $\text{PP}^{\text{NEXP}} \subseteq \text{NEXP}$. We observe that the non-determinism in the oracle NEXP calls are used in a restricted fashion; this allows us to encode the problem into exponentially many NEXP TMs, which can be simulated with a NEXP TM.

Lemma 5 *Probabilistic query entailment in Υ_{A} relative to a probabilistic database is in* NEXP *w.r.t. the combined complexity.*

With the help of Lemma 5 and Theorem 4, we conclude that for all the rule languages, the complexity of probabilistic entailment remains the same w.r.t. the combined complexity (see the second column in Table 2). Clearly, this result transfers to the case where all events are assumed to be independent. Notice, however, that the implication of Theorem 4 is stronger, as it also yields tight complexity bounds for the languages G, WG, WS, and WA w.r.t. *ba*-combined complexity, as well as for the language WG w.r.t. *fp*-combined complexity. For the remaining languages, where $\mathsf{k} = \text{NP}$, we prove Theorem 6.

Theorem 6 *If query entailment in $\mathcal{L}$ relative to databases w.r.t. ba-combined (resp., fp-combined) complexity is* NP*-complete, then probabilistic query entailment in $\Upsilon_{\mathcal{L}}$ relative to probabilistic databases is* PP^{NP}*-complete w.r.t. ba-combined (resp., fp-combined) complexity.*

Membership in PP^{NP} is shown by a setting appropriate threshold values and iterating over nondeterministic oracle calls until this threshold value is exceeded. To show hardness for this class, we require more involved technical constructions. For these constructions, we use the M-$\exists$QBF problem [41].

Definition 7 (M-$\exists$QBF) Given an integer constant c and a partially quantified Boolean formula of the form

$$\Phi = \exists y_1 \dots y_m \; \phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k,$$

where every ϕ_i is a clause over $\{x_1, \dots, x_l, y_1, \dots, y_m\}$ and $k, l, m \geq 1$; M-$\exists$QBF *(Φ,c)* is to decide whether for at least c of the truth assignments τ to $x_1, \dots, x_l$, the formula $\tau(\Phi)$ is true.

Note that M-$\exists$QBF is different from majority satisfiability, as here the threshold is set by an integer (not necessarily majority). M-$\exists$QBF is an PP^{NP}-complete problem even if the clauses ϕ_i are restricted to 3CNF [3].

The full proof constructs a probabilistic knowledge base $\mathcal{K}_\Phi = (\emptyset, \mathcal{P}_\Phi)$ and a special query Q_Φ based on Φ. $\mathcal{K}_\Phi$ and Q_Φ together simulate the satisfiability conditions for the formula Φ. Moreover, the atoms in $\mathcal{P}_\Phi$ are associated with partial assignments over the variables $\{x_1 \dots x_\ell\}$. Notice that these are precisely the variables upon which we want to decide whether the number of assignments are at least c. This construction allows us to factorize the satisfiability problem over the variables $\{x_1 \dots x_\ell\}$ and thus to obtain the result. As $\mathcal{K}_\Phi$ uses an empty program, and all atoms are bounded in the arity by 3, we obtain tight complexity bounds for all entries in Table 2.

We have analyzed the complexity of probabilistic query entailment under the standard possible world semantics. Using novel constructions, we have provided tight complexity bounds for all languages under consideration. Table 2 shows our results. Next, we provide concrete examples on how the possible world semantics can be incompetent under certain conditions, and concentrate on a different semantics.

4 INCONSISTENCY HANDLING

Due to the presence of negative constraints, knowledge bases may contain contradictory knowledge. In fact, this has lead to a quest of finding alternative semantics to be able deal with inconsistent knowledge in ontologies. Consider for example the knowledge base

$$\Sigma_{inc} = \{\mathtt{P}(x), \mathtt{R}(x) \rightarrow \bot\} \quad \text{and} \quad \mathcal{D}_{inc} = \{\mathtt{P}(u), \mathtt{R}(u), \mathtt{P}(v)\}.$$

The NC requires the predicates P and R to be disjoint, but the database states that u belongs to both of them. Thus, the program has no model relative to the database. The fact that a knowledge base contains an inconsistency makes standard reasoning very problematic, as *anything* can be entailed from an inconsistent knowledge base (*"ex falso quodlibet"*) under the standard semantics. Consequently, one loses the ability of distinguishing between queries. From a technical perspective, the inconsistency problem immediately propagates to probabilistic extensions. Consider a probabilistic variant of our example; i.e., the KB $\mathcal{K}_{inc} = (\Gamma_{inc}, \mathcal{P}_{inc})$, where

$$\Gamma_{inc} = \{\langle \mathtt{P}(x), \mathtt{R}(x) \rightarrow \bot : 0.5 \rangle\} \quad \text{and}$$
$$\mathcal{P}_{inc} = \{\langle \mathtt{P}(u) : 0.5 \rangle, \langle \mathtt{R}(u) : 0.5 \rangle, \langle \mathtt{P}(v) : 0.5 \rangle\}.$$

Observe that $\mathcal{K}_{inc}$ factorizes into worlds with positive probability that contain inconsistent knowledge. More concretely, it imposes 16 worlds, two of which are inconsistent, i.e., the ones that contain the NC together with both $\langle \mathtt{P}(u) : 0.5 \rangle$ and $\langle \mathtt{R}(u) : 0.5 \rangle$. Notice that, even though a vast majority of the worlds are consistent, the knowledge base as a whole is inconsistent, as it assigns a positive probability to an inconsistent world. It is possible to slightly change the possible world semantics to only consider consistent worlds by setting the probabilities of inconsistent worlds to 0 and renormalizing the probability distribution over the set of worlds accordingly. More precisely, assuming $\sum_{w \models \bot} \mathrm{P}_{\mathcal{K}}(w) < 1$, we obtain the distribution:

$$\mathrm{P}(w) = \begin{cases} 0 & \text{if } w \models \bot \\ \mathrm{P}_{\mathcal{K}}(w) / (1 - \sum_{w' \models \bot} \mathrm{P}_{\mathcal{K}}(w')) & \text{otherwise.} \end{cases}$$

Notice that this semantics assumes that the *error* is in the probability distribution; accordingly, it modifies the distribution. For our example, it yields a probability less than 0.5 for $\mathtt{P}(v)$. Moreover, in the same example, $\mathtt{P}(u)$ and $\mathtt{R}(u)$ evaluate to the same probability value w.r.t. this semantics. This is not in line with the intuition; particularly, because it puts as much *responsibility* on $\mathtt{P}(v)$ as much as it puts on the other tuples. On the other hand, assuming that the *error* is on the logical side, it is easy to see that responsibility needs to be shared only by the NC $\langle \mathtt{P}(x), \mathtt{R}(x) \rightarrow \bot \rangle$ and the tuples $\{\mathtt{P}(u), \mathtt{R}(u)\}$, since they serve as the source of inconsistency. Thus, it is more intuitive to expect the probability of $\mathtt{P}(v)$ to remain 0.5, as it does not contribute to inconsistency in the logical sense.

The main question is then, how to identify the meaningful answers in inconsistent worlds. We base ourselves on the recent advances on inconsistency-tolerant reasoning developed for Datalog$^{\pm}$ [28, 29, 27] and provide an inconsistency-tolerant possible world semantics. We also show that under this semantics, $\mathtt{P}(v)$ evaluates to exactly 0.5.

Several inconsistency-tolerant semantics have been proposed in the literature. One of the central semantics is first developed for relational databases [1] and then generalized as the AR semantics for several DLs [26]. The AR semantics is based on the key notion of a *repair*, which is a $\subseteq$-maximal consistent subset of the given database $\mathcal{D}$. Here, it is assumed that errors leading to inconsistencies are only contained in the data, but not in the program. In recent

work [18], authors allow errors also in the programs and introduce the *generalized repair (GR)* semantics, which allows to separate the program and the database into *hard* and *soft* parts, where the hard part is assumed to be fixed, and the soft part can be subject to repairs.

	Data	**Comb.**	ba**-comb.**	fp**-comb.**
$\mathsf{L}_\bot$, $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$	coNP	PSPACE	Π_2^p	Π_2^p
$\mathsf{G}_\bot$	coNP	2EXP	EXP	Π_2^p
$\mathsf{WG}_\bot$	EXP	2EXP	EXP	EXP
$\mathsf{S}_\bot$, $\mathsf{SF}_\bot$, $\mathsf{F}_\bot$, $\mathsf{GF}_\bot$	coNP	EXP	Π_2^p	Π_2^p
$\mathsf{A}_\bot$	coNP	P$^{\text{NEXP}}$	P$^{\text{NEXP}}$	Π_2^p
$\mathsf{WS}_\bot$, $\mathsf{WA}_\bot$	coNP	2EXP	2EXP	Π_2^p

Table 3: Complexity of GR-BQ entailment under existential rules [18]; all entries are completeness results. Hardness holds even in the case where the whole database is soft, and the whole program is hard.

Table 3 illustrates the complexity of query answering under this semantics, denoted GR-UCQ. For further details, we refer to [18]. We now extend the *generalized repair (GR)* semantics to probabilistic Datalog$^{\pm}$.

Definition 8 (Flexible programs and databases) A *flexible PDB* is a pair $\mathcal{P} = (\mathcal{P}_h; \mathcal{P}_s)$ of two PDBs $\mathcal{P}_h$ and $\mathcal{P}_s$, denoted *hard* and *soft PDB*, respectively, while a flexible (probabilistic) program is a pair $\Gamma = (\Gamma_h; \Gamma_s)$ consisting of a finite set Γ_h of TGDs and NCs and a finite set Γ_s of TGDs, denoted *hard* and *soft program*, respectively.

Consider again the probabilistic KB $\mathcal{K}_{inc} = (\Gamma_{inc}, \mathcal{P}_{inc})$, where we would like to fix the whole program, and let the whole database be a soft PDB. This can be achieved by setting $\Gamma_{inc} = (\Gamma_h, \emptyset)$ and $\mathcal{P}_{inc} = (\emptyset, \mathcal{P}_s)$, where $\Gamma_h = \Gamma_{inc}$ and $\mathcal{P}_{inc} = \mathcal{P}_s$. This partition fixes the program and views the whole PDB as a soft database. The notion of generalized repair (GR) for flexible PDBs under flexible probabilistic programs is then given as follows.

Definition 9 (Generalized repair) A *generalized repair* of a flexible PDB $(\mathcal{P}_h; \mathcal{P}_s)$ and a flexible program $(\Gamma_h; \Gamma_s)$ is a probabilistic KB $\mathcal{K} = ((\Gamma_h; \Gamma'_s), (\mathcal{P}_h; \mathcal{P}'_s))$, where $\Gamma'_s \subseteq \Gamma_s$ and $\mathcal{P}'_s \subseteq \mathcal{P}$ such that (i) $(\Gamma_h \cup \Gamma'_s \cup \mathcal{P}_h \cup \mathcal{P}'_s)$ is consistent, and (ii) there is no $t \in (\Gamma_s \cup \mathcal{P}_s)/(\Gamma'_s \cup \mathcal{P}'_s)$ such that $(\Gamma_h \cup \Gamma'_s \cup \mathcal{P}_h \cup \mathcal{P}'_s \cup \{t\})$ is consistent. The set of all such repairs is denoted by $rep(\mathcal{K})$.

Clearly, there may be many $\subset$-maximal repairs for every world: Observe that $\mathcal{K}_w = \Sigma_{inc} \cup \mathcal{D}_{inc}$ is a world over $\mathcal{K}_{inc}$. Here, both $\mathcal{K}_w/\{\langle \mathtt{P}(u)\rangle\}$ and $\mathcal{K}_w/\{\langle \mathtt{R}(u)\rangle\}$ are $\subset$-maximal repairs. In this case, these are all possible repairs. The query semantics then considers the consequences that are entailed from every $\subset$-maximal repair, i.e., the *safe consequences*.

Definition 10 (Inconsistency tolerant query semantics) Let $\mathcal{K}$ be a probabilistic KB, the *probability of a UCQ* Q is given by:

$$\mathrm{P}_{\mathcal{K}}(\mathsf{Q}) = \sum_{\mathcal{K}_{|w} \models_{\text{GR}} \mathsf{Q}} \mathrm{P}(w),$$

where $\mathcal{K}_{|w} \models_{\text{GR}} \mathsf{Q}$ holds iff for all repairs $r \in rep(\mathcal{K}_{|w})$, it holds that $r \models \mathsf{Q}$. Given a query Q and $p \in (0, 1]$, *probabilistic GR-UCQ entailment* is to decide whether $\mathrm{P}_{\mathcal{K}}(\mathsf{Q}) \geq p$.

The implication of this semantics is clear. Consider again $\mathcal{K}_{inc}$: It is easy to verify that $\mathtt{P}(v)$ is entailed from all repairs of all worlds. Therefore, it yields the probability 0.5 for $\mathtt{P}(v)$, as desired.

	Data	**Comb.**	ba**-comb.**	fp**-comb.**
$\mathsf{L}_\bot$, $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$	PP$^{\text{NP}}$	PSPACE	PP$^{\Sigma_2^p}$	PP$^{\Sigma_2^p}$
$\mathsf{G}_\bot$	PP$^{\text{NP}}$	2EXP	EXP	PP$^{\Sigma_2^p}$
$\mathsf{WG}_\bot$	EXP	2EXP	EXP	EXP
$\mathsf{S}_\bot$, $\mathsf{SF}_\bot$, $\mathsf{F}_\bot$, $\mathsf{GF}_\bot$	PP$^{\text{NP}}$	EXP	PP$^{\Sigma_2^p}$	PP$^{\Sigma_2^p}$
$\mathsf{A}_\bot$	PP$^{\text{NP}}$	in PP$^{\text{NEXP}}$	in PP$^{\text{NEXP}}$	PP$^{\Sigma_2^p}$
$\mathsf{WS}_\bot$, $\mathsf{WA}_\bot$	PP$^{\text{NP}}$	2EXP	2EXP	PP$^{\Sigma_2^p}$

Table 4: Complexity of probabilistic GR-UCQ entailment under existential rules; all entries but the "in" ones are completeness results. Hardness holds even in the case where the whole database is soft and the whole program is hard.

4.1 Complexity Results

As before, we will consider the complexity of query answering w.r.t. the different classes of probabilistic programs relative to different languages. Observe first that Theorem 4 is rather general, and thus, all the results can be transferred to this semantics.

Corollary 11 *Let* k *be the complexity of GR-UCQ entailment in* $\mathcal{L}$ *relative to a databases; then probabilistic GR-UCQ entailment in* $\Upsilon_{\mathcal{L}}$ *relative to probabilistic databases is* k*-hard,* PP*-hard, and in* PP$^{\mathsf{k}}$.

As before, this yields tight complexity bounds for languages where k is a deterministic class that contains PP. The language $\mathsf{A}_\bot$ requires a special attention, as the complexity of GR-UCQ entailment in $\mathsf{A}_\bot$ is P$^{\text{NEXP}}$-complete. We provide an upper bound for probabilistic GR-UCQ entailment.

Lemma 12 *Probabilistic GR-UCQ entailment in* Υ_{A} *relative to probabilistic databases is in* PP$^{\text{NEXP}}$ *w.r.t. the combined and* ba*-combined complexity.*

Although we expect this problem to be complete for the class PP$^{\text{NEXP}}$, it is yet open whether this problem is PP$^{\text{NEXP}}$-hard. The results for ba-combined and fa-combined cases require a much more detailed analysis. We prove the following Theorem.

Theorem 13 *Let* $\mathsf{k} = \Pi_2^p$ *be the complexity of GR-UCQ entailment in the rule language* $\mathcal{L}$ *relative to databases w.r.t.* ba*-combined (resp.,* fp*-combined) complexity. Then, probabilistic GR-UCQ entailment in* $\Upsilon_{\mathcal{L}}$ *relative to a probabilistic databases is complete in the class* PP$^{\Sigma_2^p}$ *w.r.t.* ba*-combined (resp.,* fp*-combined) complexity.*

While upper bounds can be shown using analogous arguments as in the standard semantics, to be able to show hardness, we first define a problem that is complete for the class PP$^{\Sigma_2^p}$, adopted from [41].

Definition 14 (M-$\forall\exists$QBF) Given an integer constant c and a partially quantified Boolean formula of the form

$$\Phi = \forall y_1 \dots y_m \exists z_1 \dots z_n \; \phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k \,,$$

where every ϕ_i is a clause over $\{x_1, \dots, x_l, y_1, \dots, y_m, z_1, \dots, z_n\}$ and $k, l, m, n \geq 1$; M-$\forall\exists$QBF(Φ,c) is to decide whether for at least c of the truth assignments τ to $x_1, \dots, x_l$, the formula $\tau(\Phi)$ is true.

For an arbitrary instance of M-$\forall\exists$QBF, where the clauses are restricted to 3CNF, we construct a probabilistic knowledge base that consists of a program that contains a single negative constraint. The reduction is correct and of polynomial size. As the KB provided is

in the intersection of the class of languages considered, we obtain hardness for all of these languages.

Our last result is about data complexities, for which GR-UCQ entailment is coNP-complete (except for the class $\mathsf{WG}_{\perp}$, which has exponential data complexity). We prove the following Theorem.

Theorem 15 *If GR-UCQ entailment in $\mathcal{L}$ relative to databases is co*NP*-complete (or* NP*-complete) in data complexity, then the data complexity of probabilistic GR-UCQ entailment in $\Upsilon_{\mathcal{L}}$ relative to probabilistic databases is* $\mathrm{PP}^{\mathrm{NP}}$*-complete.*

We use the canonical problem M-$\exists$QBF to show hardness. Observe, however, that this requires a very different construction than the one in Theorem 6, as here the query is fixed. This result concludes our complexity analysis for inconsistency-tolerant semantics; all of the results are summarized in Table 4.

5 RELATED WORK

Our work is closely related to probabilistic databases [37, 12]. In fact, probabilistic Datalog$^{\pm}$ can be seen as a generalization of PDBs. As in PDBs, query answering in probabilistic Datalog$^{\pm}$ is PP-hard. Notice the novel dichotomy result in tuple-independent PDBs that classifies the unions of conjunctive queries as either being safe (P), or unsafe (#P-hard) [13]. Identifying special cases (as in [24]) of probabilistic Datalog$^{\pm}$ that allows similar dichotomies is part of the future work. We also note the recent work on Open-World PDBs [10], which allows a more flexible representation for PDBs by providing default probability intervals for unknown facts; significantly, this extended setting preserves the full dichotomy result for UCQs.

The possible world semantics is widely employed in probabilistic logic programming [34, 25, 35], and probabilistic query answering has been studied in light-weight probabilistic ontology languages [11, 14, 24] before; see especially [30] for an overview of probabilistic ontology languages. Our approach differs in several aspects: First, we consider a family of existential rule languages that is known to be well-behaved and provide tight complexity bounds for reasoning in these languages for all of the cases except one. Second, our assumptions are rather flexible, as we do not require a specific probabilistic model. Lastly, we propose an inconsistency-tolerant semantics, based on [18], and study query evaluation under this semantics. Note that inconsistency-tolerant semantics have been studied in Datalog$^{\pm}$ [21] before, which we find closely related. Differently, we adopt a more general repair semantics, as we allow repairs both on the data and on the program, and it is possible to partition the knowledge base into hard and soft components. Finally, we consider the full Datalog$^{\pm}$ family (not only guarded rules), providing a complete picture of the complexity of query evaluation.

6 SUMMARY AND OUTLOOK

We have studied probabilistic query entailment in Datalog$^{\pm}$ under the standard possible world semantics and under an inconsistency-tolerant variant of it. We have shown that the inconsistency-tolerant semantics provides more information, while pushing the computational complexity of probabilistic query entailment higher in the counting polynomial-time hierarchy in many cases. The differences between these two semantic considerations represent yet another trade-off between retrieving more information on the one side and the increasing computational cost on the other side. Our analysis is purely complexity-theoretical, and it is an open research problem to find special cases where efficient algorithms can be developed. Such algorithms can take the advantage of existing methods in knowledge compilation [15, 7], as performing operations on a pre-compiled structure is known to be very efficient.

ACKNOWLEDGEMENTS

This work was supported by the German Research Foundation (DFG) within RoSI (GRK 1907) and by the UK EPSRC grants EP/J008346/1, EP/L012138/1, and EP/M025268/1.

A PROOF SKETCH FOR THEOREM 4

We prove the result only w.r.t. the data complexity; the result w.r.t. the *ba*-combined, *fa*-combined, and combined complexity can be obtained using analogous arguments.

(Hardness) Let k be the data complexity of query entailment in $\mathcal{L}$. Probabilistic query entailment w.r.t. the data complexity is PP-hard, as it is already so in PDBs w.r.t. data complexity. Thus, we only need to show k-hardness w.r.t. data complexity. Suppose that probabilistic query entailment is not k-hard in $\Upsilon_{\mathcal{L}}$ w.r.t. the data complexity. Let Σ be an arbitrary program relative to an arbitrary database $\mathcal{D} = \{l_i \mid 1 \leq i \leq n\}$ and construct the probabilistic KB $\mathcal{K}' = (\Gamma', \mathcal{P}')$ where

$$\Gamma' = \{\lambda \mid \lambda \in \Sigma\}, \;\; \mathcal{P}' = \{\langle l_i : 0.5\rangle \mid l_i \in \mathcal{D}, 1 \leq i \leq n\}\,.$$

Clearly, this construction is polynomial, and given a query Q', it is easy to see that

$$(\Sigma, \mathcal{D}) \models \mathsf{Q}' \text{ holds iff } \mathrm{P}(\mathsf{Q}') > 0.5^n,$$

which implies that query answering in $\mathcal{L}$ is not k-hard in the data complexity, which leads to a contradiction.

(Membership) We assume that the probability of each world is computable in polynomial time, that it is a rational number, and that the rational numbers of the probabilities of all worlds have the same denominator. As for membership in PP^{k}, intuitively, we first create multiples of each world (which then correspond to the nondeterministic branches of a Turing machine), so that the probability distribution over all thus generated worlds is the uniform distribution. Then, for thresholds properly below (resp., above) 0.5, we introduce artificial success (resp., failure) worlds (which correspond to other nondeterministic success (resp., failure) branches of a Turing machine), so that satisfying the resulting threshold corresponds to having a majority of success worlds. We thus only have to verify whether for the majority of the worlds, the query evaluates to true. As query evaluation is in k, the computation is overall in PP^{k}. □

B PROOF SKETCH FOR LEMMA 5

Let Q be a UCQ, and $\mathcal{K} = (\Gamma, \mathcal{P})$ be an arbitrary probabilistic knowledge base where Γ is defined over Υ_{A}. By Definition 3, it suffices to decide whether $\sum_{\mathcal{K}_{|w} \models \mathsf{Q}} \mathrm{P}(w) \geq p$. Let W be the set of all worlds w. Guess a subset $\{w_1, \ldots, w_n\} \subseteq W$ and verify whether

$$(1) \sum_{\mathcal{K}_{|w_i}} \models \mathsf{Q} \text{ for all } \{w_i \mid 1 \leq i \leq n\} \text{ and } \quad (2) \sum_{1}^{n} \mathrm{P}(w_i) \geq p.$$

It is easy to see that this procedure yields the correct decision. We only need to show that this procedure is in NEXP. First, observe

that the guess is of size exponential and can be produced by a nondeterministic Turing machine that runs in exponential time. Second, the verification of (1) can be done in (EXP $\times$ NEXP) = NEXP, as there are exponentially many worlds and k = NEXP. Finally, the verification of (2) can be done by traversing over exponentially many worlds and computing their probabilities. As the latter can be done in polynomial time, this verification is clearly in EXP. □

C PROOF SKETCH FOR THEOREM 6

(Membership) This result is a consequence of Theorem 4, where k is set to NP.

(Hardness) To show hardness, we provide a reduction from the M-∃QBF problem (Definition 7). Let the formula in 3CNF

$$\Phi = \exists y_1 \dots y_m \ \phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k,$$

be a partially quantified Boolean formula defined over $V = \{x_1, \dots, x_l, y_1, \dots, y_m\}$, where every ϕ_i is a disjunction of three literals, and c is an integer constant. For every clause $\phi_i = \circ u_i \vee \circ v_i \vee \circ w_i$, define ground atoms $\mathtt{M_i}(\nu_i(u_i), \nu_i(v_i), \nu_i(w_i))$, where ν_i is a truth assignment to the variables u_i, v_i, w_i that satisfies ϕ_i. Observe that, for every clause, the number of such assignments is bounded by 2^3. The partial assignment $\nu_{|s_1 \dots s_n}$ denotes the restriction of ν to the variables $\{s_1, \dots, s_n\}$. We construct the probabilistic KB $\mathcal{K}_\Phi = (\emptyset, \mathcal{P}_\Phi)$, where

$$\mathcal{P}_\Phi = \{\langle \mathtt{M_i}(\nu_i(u_i), \nu_i(v_i), \nu_i(w_i)) \colon \nu_{|\{x_1,\dots,x_l\} \cap \{u_i,v_i,w_i\}} \rangle \mid \nu_i \models \phi_i, 1 \le i \le k\}.$$

Let the event space be defined over the x-variables such that every world has the probability 0.5^l. For the query

$$\mathsf{Q}_\Phi = \exists\, x_1 \dots x_l, y_1 \dots y_m \ \bigwedge_{i=1}^{k} \mathtt{M}_i(u_i, v_i, w_i),$$

we obtain the following reduction:

$$\mathrm{P}_{\mathcal{K}_\Phi}(\mathsf{Q}) \ge \mathsf{c} \cdot 0.5^l \text{ iff M-}\exists\text{QBF}(\Phi, \mathsf{c}) \text{ answers yes.}$$

Observe that the above reduction can clearly be done in polynomial time in the size of Φ, and that the resulting probabilistic program is empty, and the arity of all predicates in the PDB is 3. □

D PROOF SKETCH FOR THEOREM 13

(Membership) This result is a consequence of Theorem 4 and Corollary 11, where k is set to Π_2^p, and the fact that $\mathrm{PP}^{\Sigma_2^p} = \mathrm{PP}^{\Pi_2^p}$.

(Hardness) We provide a reduction from an arbitrary instance of M-∀∃QBF given in Definition 14. Let

$$\Phi = \forall y_1 \dots y_m \exists z_1 \dots z_n \ \phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_k,$$

be a partially quantified Boolean formula in 3CNF over $V = \{x_1, \dots, x_l, y_1, \dots, y_m, z_1, \dots, z_n\}$, where every clause ϕ_i is a disjunction of three literals, and c is an integer constant. For every clause $\phi_i = \circ u_i \vee \circ v_i \vee \circ w_i$, define ground atoms $\mathtt{M_i}(\nu_i(u_i), \nu_i(v_i), \nu_i(w_i))$, where ν_i is a truth assignment to the variables u_i, v_i, w_i that satisfies ϕ_i. Observe that, for every clause, the number of such assignments is bounded by 2^3. The partial assignment $\nu_{|s_1 \dots s_n}$ denotes the restriction of ν to the variables $\{s_1, \dots, s_n\}$. Construct the probabilistic KB $\mathcal{K}_\Phi = (\Gamma_\Phi, \mathcal{P}_\Phi)$ where

$$\mathcal{P}_\Phi = \{\langle \mathtt{M_i}(\nu_i(u_i), \nu_i(v_i), \nu_i(w_i)) \colon \nu_{|\{x_1,\dots,x_l\} \cap \{u_i,v_i,w_i\}} \rangle \mid \nu_i \models \phi_i, 1 \le i \le k\} \cup \{\langle \mathtt{S}(0,1,i) \rangle, \langle \mathtt{S}(1,0,i) \rangle \mid 1 \le i \le m\},$$

$$\Gamma_\Phi = \{\mathtt{S}(x,y,z) \wedge \mathtt{S}(y,x,z) \to \bot\}.$$

Here, all probabilistic facts are soft, and the NC is hard. Let the event space be defined over the x-variables such that every world has the probability 0.5^l. Then, for the query

$$\mathsf{Q}_\Phi = \exists\, x_1 \dots x_l, y_1 \dots y_m \ z_1 \dots z_l \ \bigwedge_{i=1}^{k} \mathtt{M}_i(u_i, v_i, w_i) \wedge \bigwedge_{i=1}^{m} \mathtt{S}(y_i, y_i', i),$$

we obtain the reduction:

$$\mathrm{P}_{\mathcal{K}_\Phi}(\mathsf{Q}) \ge \mathsf{c} \cdot 0.5^l \text{ iff M-}\exists\text{QBF}(\Phi, \mathsf{c}) \text{ answers yes.}$$

Observe that the above reduction can be done in polynomial time in the size of Φ, that the resulting probabilistic program is fixed and consists only of one NC, and the arity of all predicates is at most 3. □

E PROOF SKETCH FOR THEOREM 15

(Membership) This result is a consequence of Theorem 4 and Corollary 11, where k is set to coNP, and the fact that $\mathrm{PP}^{\mathrm{coNP}} = \mathrm{PP}^{\mathrm{NP}}$

(Hardness) We provide a reduction from the $\mathrm{PP}^{\mathrm{NP}}$-complete problem of, given a partially quantified Boolean formula

$$\Phi = \forall y_1 \dots y_m \ \phi_1 \vee \phi_2 \vee \dots \vee \phi_k,$$

over $V = \{x_1, \dots, x_l, y_1, \dots, y_m\}$, where every ϕ_i is a conjunction of three literals, and an integer constant c, deciding whether for at least c truth assignments τ to $x_1 \dots x_l$, the formula

$$\forall y_1 \dots y_m \ \tau(\phi_1) \vee \tau(\phi_2) \vee \dots \vee \tau(\phi_k)$$

is true. Let $\phi_i = \circ u_{i,1} \wedge \circ u_{i,2} \wedge \circ u_{i,3}$. We define the PDB $\mathcal{P}_\Phi$ that contains the deterministic tuples

$$\langle \mathtt{M}(u_{i,1}, u_{i,1}', u_{i,2}, u_{i,2}', u_{i,3}, u_{i,3}', i) \rangle$$

such that $u_{i,j}' = 1$ (resp., $u_{i,j}' = 0$), if $u_{i,j}$ occurs positively (resp., negatively) in ϕ_i, $1 \le i \le k$, $1 \le j \le 3$. Furthermore, we add the soft probabilistic facts $\langle \mathtt{S}(0,1,x_i) \colon \neg x_i \rangle$ and $\langle \mathtt{S}(1,0,x_i) \colon x_i \rangle$ such that $i \in \{1, \dots, l\}$, and the soft probabilistic facts $\langle \mathtt{S}(0,1,y_i) \rangle$ and $\langle \mathtt{S}(1,0,y_i) \rangle$ such that $i \in \{1, \dots, m\}$.

We define the program $\Gamma_\Phi = \{\mathtt{S}(x,y,z) \wedge \mathtt{S}(y,x,z) \to \bot\}$, consisting of one hard NC, and therewith the probabilistic KB $\mathcal{K}_\Phi = (\Gamma_\Phi, \mathcal{P}_\Phi)$. The event space is defined over the x-variables such that every world has the probability 0.5^l. Then, for the query Φ

$$\exists \dots \mathtt{M}(u_i, a_i, v_i, b_i, w_i, c_i, i) \wedge \mathtt{S}(a_i, a_i', u_i) \wedge \mathtt{S}(b_i, b_i', v_i) \wedge \mathtt{S}(c_i, c_i', w_i),$$

we obtain that $\mathrm{P}_{\mathcal{K}_\Phi}(\mathsf{Q}) \ge \mathsf{c} \cdot 0.5^l$ iff for at least c truth assignments τ to $x_1 \dots x_l$, the formula

$$\forall y_1 \dots y_m \ \tau(\phi_1) \vee \tau(\phi_2) \vee \dots \vee \tau(\phi_k)$$

is true. This reduction can clearly be done in polynomial time in the size of Φ, the resulting probabilistic program consists of exactly one NC and is fixed, and the query is also fixed. □

REFERENCES

[1] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki, 'Consistent query answers in inconsistent databases', in *Proc. PODS*, pp. 68–79. ACM Press, (1999).

[2] *The Description Logic Handbook: Theory, Implementation, and Applications*, eds., Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, Cambridge University Press, 2nd edn., 2007.

[3] Delbert D. Bailey, Víctor Dalmau, and Phokion G. Kolaitis, 'Phase transitions of PP-complete satisfiability problems', *Discrete Appl. Math.*, **155**(12), 1627–1639, (2007).

[4] Catriel Beeri and Moshe Y. Vardi, 'The implication problem for data dependencies', in *Proc. ICALP*, volume 115 of *LNCS*, pp. 73–85. Springer, (1981).

[5] Meghyn Bienvenu, 'On the complexity of consistent query answering in the presence of simple ontologies', in *Proc. AAAI*. AAAI Press, (2012).

[6] Meghyn Bienvenu and Riccardo Rosati, 'Tractable approximations of consistent query answering for robust ontology-based data access', in *Proc. IJCAI*, (2013).

[7] Guy Van Den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt, 'Lifted probabilistic inference by first-order knowledge compilation', in *Proc. IJCAI*, pp. 2178–2185, (2011).

[8] Andrea Calì, Georg Gottlob, and Michael Kifer, 'Taming the infinite chase: Query answering under expressive relational constraints', *J. Artif. Intell. Res.*, **48**, 115–174, (2013).

[9] Andrea Calì, Georg Gottlob, and Andreas Pieris, 'Towards more expressive ontology languages: The query answering problem', *Artif. Intell.*, **193**, 87–128, (2012).

[10] İsmail İlkan Ceylan, Adnan Darwiche, and Guy Van Den Broeck, 'Open-world probabilistic databases', in *Proc. KR*, pp. 339–348. AAAI Press, (2016).

[11] İsmail İlkan Ceylan and Rafael Peñaloza, 'Probabilistic query answering in the Bayesian description logic $\mathcal{BEL}$', in *Proc. SUM*, volume 9310 of *LNAI*, pp. 21–35. Springer, (2015).

[12] Nilesh Dalvi, Christopher Ré, and Dan Suciu, 'Probabilistic databases: diamonds in the dirt', *Commun. ACM*, **52**(7), 86–94, (2009).

[13] Nilesh Dalvi and Dan Suciu, 'The dichotomy of probabilistic inference for unions of conjunctive queries', *J. ACM*, **59**(6), 1–87, (2012).

[14] Claudia D'Amato, Nicola Fanizzi, and Thomas Lukasiewicz, 'Tractable reasoning with Bayesian description logics', in *Proc. SUM*, volume 5291 of *LNAI*, pp. 146–159. Springer, (2008).

[15] Adnan Darwiche and Pierre Marquis, 'A knowledge compilation map', *J. Artif. Intell. Res.*, **17**, 229–264, (2011).

[16] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen, 'ProbLog: A probabilistic Prolog and its application in link discovery', in *Proc. IJCAI*, pp. 2468–2473. Morgan-Kaufmann, (2007).

[17] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Patrick Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang, 'Knowledge Vault: A Web-scale approach to probabilistic knowledge fusion', in *Proc. KDD*, pp. 601–610. ACM Press, (2014).

[18] Thomas Eiter, Thomas Lukasiewicz, and Livia Predoiu, 'Generalized consistent query answering under existential rules', in *Proc. KR*. AAAI Press, (2016).

[19] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa, 'Data exchange: Semantics and query answering', *Theor. Comput. Sci.*, **336**(1), 89–124, (2005).

[20] John T. Gill, 'Computational complexity of probabilistic Turing machines', *SIAM J. Comput.*, **6**(4), 675–695, (1977).

[21] Georg Gottlob, Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari, 'Query answering under probabilistic uncertainty in Datalog+/– ontologies', *Ann. Math. Artif. Intell.*, **69**(1), 37–72, (2013).

[22] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum, 'YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia', in *Proc. IJCAI*, pp. 3161–3165, (2013).

[23] Tomasz Imieliski and Witold Lipski, 'Incomplete information in relational databases', *J. ACM*, **31**(4), 761–791, (1984).

[24] Jean Christoph Jung and Carsten Lutz, 'Ontology-based access to probabilistic data with OWL QL', in *Proc. ISWC*, volume 7649 of *LNCS*, pp. 182–197. Springer, (2012).

[25] Marta Kwiatkowska, Gethin Norman, and David Parker, 'PRISM: Probabilistic symbolic model checker', *Computer Performance Evaluation Modelling Techniques and Tools*, **2324**, 200–204, (2002).

[26] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo, 'Inconsistency-tolerant semantics for description logics', in *Proc. RR*, pp. 103–117, (2010).

[27] Thomas Lukasiewicz, Maria Vanina Martinez, Andreas Pieris, and Gerardo I. Simari, 'From classical to consistent query answering under existential rules', in *Proc. AAAI*, pp. 1546–1552. AAAI Press, (2015).

[28] Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari, 'Inconsistency handling in Datalog+/– ontologies', in *Proc. ECAI*, pp. 558–563, (2012).

[29] Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari, 'Complexity of inconsistency-tolerant query answering in Datalog+/–', in *Proc. ODBASE*, pp. 488–500, (2013).

[30] Thomas Lukasiewicz and Umberto Straccia, 'Managing uncertainty and vagueness in description logics for the Semantic Web', *J. Web Sem.*, **6**(4), 291–308, (2008).

[31] Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Tanti Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling, 'Never-ending learning', in *Proc. AAAI*, pp. 2302–2310. AAAI Press, (2015).

[32] Thomas Rölleke Norbert Fuhr, 'A probabilistic relational algebra for the integration of information retrieval and database systems', *ACM Trans. Inf. Syst.*, **15**(1), 32–66, (1997).

[33] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati, 'Linking data to ontologies', *J. Data Sem.*, **10**, 133–173, (2008).

[34] David Poole, 'The independent choice logic for modelling multiple agents under uncertainty', *Artif. Intell.*, **94**(1-2), 7–56, (1997).

[35] Joris Renkens, Dimitar Shterionov, Guy Van den Broeck, Jonas Vlasselaer, Daan Fierens, Wannes Meert, Gerda Janssens, and Luc De Raedt, 'ProbLog2: From probabilistic programming to statistical relational learning', *Proc. NIPS*, 1–5, (2012).

[36] Jaeho Shin, Feiran Wang, Christopher De Sa, Ce Zhang, and Sen Wu, 'Incremental knowledge base construction using DeepDive', in *Proc. VLDB*, volume 8, (2015).

[37] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. Probabilistic Databases, 2011.

[38] Jacobo Torán, 'Complexity classes defined by counting quantifiers', *J. ACM*, **38**(3), 753–774, (1991).

[39] Leslie Gabriel Valiant, 'The complexity of computing the permanent', *Theor. Comput. Sci.*, **8**(2), 189–201, (1979).

[40] Moshe Y. Vardi, 'The complexity of relational query languages', *J. ACM*, 137–146, (1982).

[41] Klaus W. Wagner, 'The complexity of combinatorial problems with succinct input representation', *Acta Inf.*, **23**(3), 325–356, (1986).

[42] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu, 'Probase: A probabilistic taxonomy for text understanding', *Proc. SIGMOD*, 481–492, (2012).