Proceedings, Fifteenth International Conference on
Principles of Knowledge Representation and Reasoning (KR 2016)

Open-World Probabilistic Databases

İsmail İlkan Ceylan and **Adnan Darwiche** and **Guy Van den Broeck**
Computer Science Department
University of California, Los Angeles
ceylan@tcs.inf.tu-dresden.de,{darwiche,guyvdb}@cs.ucla.edu

Abstract

Large-scale probabilistic knowledge bases are becoming increasingly important in academia and industry alike. They are constantly extended with new data, powered by modern information extraction tools that associate probabilities with database tuples. In this paper, we revisit the semantics underlying such systems. In particular, the closed-world assumption of probabilistic databases, that facts not in the database have probability zero, clearly conflicts with their everyday use. To address this discrepancy, we propose an open-world probabilistic database semantics, which relaxes the probabilities of open facts to intervals. While still assuming a finite domain, this semantics can provide meaningful answers when some probabilities are not precisely known. For this open-world setting, we propose an efficient evaluation algorithm for unions of conjunctive queries. Our open-world algorithm incurs no overhead compared to closed-world reasoning and runs in time linear in the size of the database for tractable queries. All other queries are #P-hard, implying a data complexity dichotomy between linear time and #P. For queries involving negation, however, open-world reasoning can become NP-, or even NP^{PP}-hard. Finally, we discuss additional knowledge-representation layers that can further strengthen open-world reasoning about big uncertain data.

1 Introduction

Driven by the need to learn from vast amounts of text data, efforts throughout natural language processing, information extraction, databases and AI are coming together to build *large-scale knowledge bases*. Academic systems such as NELL (Mitchell et al. 2015), Reverb (Fader, Soderland, and Etzioni 2011), Yago (Hoffart et al. 2013), and DeepDive (Shin et al. 2015) continuously crawl the web to extract relational information. Industry projects such as Microsoft's Probase (Wu et al. 2012) or Google's Knowledge Vault (Dong et al. 2014) similarly learn structured data from text to improve search products. These systems have already populated their databases with millions of entities and billions of tuples.

Such knowledge bases are inherently probabilistic. To go from the raw text to structured data, information extraction systems employ a sequence of statistical machine learning techniques, from part-of-speech tagging until relation extraction (Mintz et al. 2009). For knowledge-base completion – the task of deriving new facts from existing knowledge – statistical relational learning algorithms make use of embeddings (Bordes et al. 2011; Socher et al. 2013) or probabilistic rules (Wang, Mazaitis, and Cohen 2013; De Raedt et al. 2015). In both settings, the output is a predicted fact with its probability. It is therefore required to define probabilistic semantics for knowledge bases. The classical and most-basic model is that of *tuple-independent probabilistic databases* (PDBs) (Suciu et al. 2011), which indeed underlies many of these systems (Dong et al. 2014; Shin et al. 2015). According to the PDB semantics, each database tuple is an independent Bernoulli random variable, and all other tuples have probability zero, enforcing a *closed-world assumption* (CWA) (Reiter 1978).

This paper revisits the choice for the CWA in probabilistic knowledge bases. We observe that the *CWA is violated* in their deployment, which makes it *problematic to reason, learn, or mine* on top of these databases. We will argue the following salient points in detail. First, knowledge bases are part of a larger machine learning loop that continuously updates beliefs about facts based on new textual evidence. From a Bayesian learning perspective (Bishop 2006), this loop can only be principled when learned facts have an a priori non-zero probability. Hence, the CWA does not accurately represent this mode of operation and puts it on weak footing. Second, these issues are not temporary: it will never be possible to complete probabilistic knowledge bases of even the most trivial relations, as the memory requirements quickly become excessive. This already manifests today: statistical classifiers output facts at a high rate, but only the most probable ones make it into the knowledge base, and the rest is truncated, losing much of the statistical information. For example, 99% of the tuples in NELL have a probability larger than 0.91. Third, query answering under the CWA does not take into account the effect the open world can have on the query probability. This makes it impossible to distinguish queries whose probability should intuitively differ. Finally, these issues stand in the way of some principled approaches to knowledge base completion and mining.

We propose an alternative semantics for probabilistic knowledge bases to address these problems, based on the *open-world assumption (OWA)*. OWA does not presume that

the knowledge of a domain is complete. Hence, anything that is not in the DB remains possible. Our proposal for *open-world probabilistic databases* (OpenPDBs) builds on the theory of imprecise probabilities, and credal sets in particular (Levi 1980), to allow interval-based probabilities for open tuples. In the most-basic setting, OpenPDBs make explicit the probability threshold that decides which facts make it into the knowledge base. All facts in the open world must have a lower probability, which bounds their contribution to the probability of possible worlds. This framework provides more meaningful answers, in terms of upper and lower bounds on the query probability. Throughout this paper, we assume a fixed and finite domain. Probabilistic reasoning with an unknown number of objects is an important problem that comes with its own challenges (Milch et al. 2007), which we leave for future work.

Our open-world semantics is supported by a *query evaluation algorithm* for *unions of conjunctive queries* (UCQs). This class of queries, corresponding to monotone DNF, is particularly well-behaved and the focal point of database research. Perhaps the largest appeal of PDBs comes from a breakthrough dichotomy result by Dalvi and Suciu (2012), perfectly delineating which UCQs can be answered efficiently in the size of the PDB. Their algorithm runs in polynomial time for all efficient queries, called *safe queries*, and recognizes all others to be #P-hard. Our OpenPDB algorithm extends the PDB algorithm of Dalvi and Suciu (2012) and inherits its elegant properties: all safe queries run in polynomial time. When our algorithm fails, the query is #P-hard. Moreover, a careful analysis shows that both algorithms run in *linear time* in the number of (closed-world) tuples. Even though OpenPDBs model a polynomially larger set of random variables, these can be reasoned about as a whole, and there is no computational blow-up for open-world reasoning. Hence, both OpenPDBs and PDBs admit the same *data complexity dichotomy between linear time and #P*.[1]

For queries with negation, only a partial classification of PDB data complexity is known (Gribkoff, Van den Broeck, and Suciu 2014; Fink and Olteanu 2015). We show that the complexity of open-world reasoning can go up significantly with negation. We identify a linear-time PDB query that becomes NP-complete on OpenPDBs. Moreover, there exists a PP-complete[2] query on PDBs that becomes NP^{PP}-complete on OpenPDBs. Clearly, negation leads to a much richer data complexity landscape. We also consider query evaluation complexity in terms of the domain size, or equivalently, the size of the open world, keeping both the query and the database fixed. Here, complexities range from polynomial time to the unary alphabet class $\#\text{P}_1$. Finally, we pinpoint limitations of OpenPDBs in their basic form and look at promising directions for this framework. For the proofs of theorems and lemmas, we refer to the extended version of this paper.[3]

[1] The fact that safe PDB queries have linear-time data complexity, and that the dichotomy of Dalvi and Suciu (2012) is between linear time (not PTime) and #P is perhaps not technically surprising. However, this has not been observed in the literature as far as we know. This observation is quite important practically though, particularly in the context of open-world probabilistic databases.

[2] PP is the decision version of the counting class #P.

Inmovie	
w_smith	ali
arquette	scream
pitt	mr_ms_smith
jolie	mr_ms_smith

Couple	
arquette	cox
pitt	jolie
pitt	aniston
kunis	kutcher

Figure 1: Database tables. Each row is interpreted as an atom. For example, the first row in the left table is interpreted as atom $\texttt{Inmovie}(\mathsf{w\_smith}, \mathsf{ali})$.

2 Closed-World Databases

Before introducing OpenPDBs we review classical and probabilistic databases, and discuss the implications of the closed-world assumption for these representations.

2.1 Relational Logic

Throughout this paper, we will work with the *function-free finite-domain* fragment of first-order logic (FOL). An *atom* $\texttt{P}(t_1, \ldots, t_n)$ consists of predicate $\texttt{P}/n$ of arity n followed by n arguments, which are either *constants* from a finite domain $\mathcal{D} = \{\mathsf{a}, \mathsf{b}, \ldots\}$ or *logical variables* $\{x, y, \ldots\}$. A *ground atom* does not contain logical variables. A *literal* is an atom or its negation. A *formula* combines atoms with logical connectives and quantifiers $\exists$ and $\forall$. A logical variable x is *quantified* if it is enclosed by a $\forall x$ or $\exists x$. A *free variable* is one that is not quantified. We write $\phi(x, y)$ to denote that x, y are free in ϕ. A *clause* is a disjunction of literals and a *CNF* is a conjunction of clauses. A *term* is a conjunction of literals and a *DNF* is a disjunction of terms. A formula is *monotone* if it contains no negations. A *substitution* $[x/t]$ replaces all occurrences of x by t in some formula Q, denoted $\mathsf{Q}[x/t]$.

A relational *vocabulary* σ consists of a set of predicates $\mathcal{R}$ and a domain $\mathcal{D}$. We will make use of *Herbrand semantics* (Hinrichs and Genesereth 2006), as is customary. The *Herbrand base* of σ is the set of all ground atoms that can be constructed from $\mathcal{R}$ and $\mathcal{D}$. An *σ-interpretation* is a truth-value assignment to all the atoms in the Herbrand base of σ, called σ-atoms. An interpretation ω is a model of formula Q when it satisfies Q, defined in the usual way. Satisfaction is denoted by $\omega \models_\sigma \mathsf{Q}$. We omit σ from notation when it is clear from context.

2.2 Databases and Queries

Following the standard model-theoretic view (Abiteboul, Hull, and Vianu 1995), a *relational database* for vocabulary σ is a σ-interpretation ω. Figure 1 depicts a classical representation of a relational database in terms of tables. Each table corresponds to a predicate and its rows correspond to ground atoms of that predicate, which are also called *records* or *facts*. These atoms are mapped to *true*, while ones not listed in these tables are mapped to *false,* according to the *closed-world assumption* (CWA) (Reiter

[3] Available at http://web.cs.ucla.edu/~guyvdb/

Inmovie		P
w_smith	ali	0.9
w_smith	sharktale	0.8
j_smith	ali	0.6
arquette	scream	0.7
pitt	mr_ms_smith	0.5
jolie	mr_ms_smith	0.7
jolie	sharktale	0.9

Couple		P
arquette	cox	0.6
pitt	jolie	0.8
thornton	jolie	0.6
pitt	aniston	0.9
kunis	kutcher	0.7

Figure 2: Probabilistic database tables. Each row is interpreted as a tuple $\langle t : p \rangle$, where t is an atom and p is a probability. For example, the first row in the left table is interpreted as the tuple $\langle \texttt{Inmovie}(\texttt{w\_smith}, \texttt{ali}) : 0.9 \rangle$.

1978). A σ-interpretation ω is also sometimes represented as a set that contains all ground atoms mapped to *true*. The database in Figure 1 will then be represented by the set $\{\texttt{Inmovie}(\texttt{w\_smith}, \texttt{ali}), \ldots, \texttt{Couple}(\texttt{kunis}, \texttt{kutcher})\}$. We will adopt this set notation in the following definitions.

The fundamental task in databases is query answering. Given a formula $\mathsf{Q}(x, y, \ldots)$, the task is to find all substitutions (answers) $[x/s, y/t, \ldots]$ such that $\omega \models \mathsf{Q}[x/s, y/t, \ldots]$. That is, find assignments to the free variables to satisfy the query. Consider for example the query $\mathsf{Q}_1(x, y)$ for spouses that starred in the same movie:

$$\exists z,\ \texttt{Inmovie}(x, z) \wedge \texttt{Inmovie}(y, z) \wedge \texttt{Couple}(x, y).$$

The database in Figure 1 yields $[x/\texttt{pitt}, y/\texttt{jolie}]$ as the only answer. This formula is an existentially quantified conjunction of atoms (i.e., no negation), called a *conjunctive query* (CQ). A common notation for this query is

$$\mathsf{Q}_1(x, y) = \texttt{Inmovie}(x, z), \texttt{Inmovie}(y, z), \texttt{Couple}(x, y).$$

We concentrate on *Boolean conjunctive queries* (BCQs), which have no free variables. Answers to BCQs are either *true* or *false*. For example, the BCQ

$$\mathsf{Q}_2 = \texttt{Inmovie}(x, z), \texttt{Inmovie}(y, z), \texttt{Couple}(x, y),$$

where all variables are existentially quantified, returns *true* on the database in Figure 1. A Boolean *union of conjunctive queries* (UCQ) is a disjunction of BCQs. From a logical perspective UCQs are motonone ($\exists$-)DNF sentences. While it is common to restrict queries to monotone fragments, it can be useful to allow negation in database queries. We will denote the class of *UCQs with negation on atoms* by UC$\tilde{\text{Q}}$, corresponding to non-monotone existentially-quantified DNF. BCQs with negation are denoted by BC$\tilde{\text{Q}}$.

2.3 Probabilistic Databases

The simplest probabilistic database model is the one based on the tuple-independence assumption. We adopt this model in this paper and refer to Suciu et al. (2011) for details on this model and other alternatives.

Definition 1. A *probabilistic database* (PDB) $\mathcal{P}$ for a vocabulary σ is a finite set of *tuples* of the form $\langle t : p \rangle$, where t is a σ-atom and $p \in [0, 1]$. Moreover, if $\langle t : p \rangle \in \mathcal{P}$ and $\langle t : q \rangle \in \mathcal{P}$, then $p = q$.

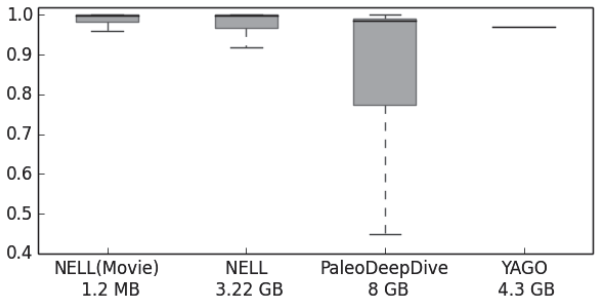

Figure 3: Box plot of probabilistic knowledge base probabilities with 2nd to 3rd quartile in gray. YAGO only provides an estimate of the mean probability per relation.

Figure 2 shows an example PDB. The following semantics is based on the *tuple-independence* assumption mentioned earlier (Suciu et al. 2011).

Definition 2. A PDB $\mathcal{P}$ for vocabulary σ induces a *unique probability distribution* over σ-interpretations ω:

$$\mathrm{P}_{\mathcal{P}}(\omega) = \prod_{t \in \omega} \mathrm{P}_{\mathcal{P}}(t) \prod_{t \notin \omega} (1 - \mathrm{P}_{\mathcal{P}}(t)),$$

where

$$\mathrm{P}_{\mathcal{P}}(t) = \begin{cases} p & \text{if } \langle t : p \rangle \in \mathcal{P} \\ 0 & \text{otherwise.} \end{cases}$$

The choice of setting $\mathrm{P}_{\mathcal{P}}(t) = 0$ for tuples missing from PDB $\mathcal{P}$ is a probabilistic version of the CWA.

Consider now the PDB $\mathcal{P}$ given in Figure 2, and the σ-interpretation ω given in Figure 1, $\{\texttt{Inmovie}(\texttt{w\_smith}, \texttt{ali}), \ldots, \texttt{Couple}(\texttt{kunis}, \texttt{kutcher})\}$. We then have $\mathrm{P}(\omega) = 0.9 \cdot (1 - 0.8) \cdots (1 - 0.6) \cdot 0.9 \cdot 0.7$. Note, however, that if we add the fact $\texttt{Couple}(\texttt{w\_smith}, \texttt{j\_smith})$ to ω, then $\mathrm{P}(\omega) = 0$ because the additional fact has no corresponding tuple in the PDB $\mathcal{P}$.

Query languages remain the same in PDBs, except that we can now compute their probabilities.

Definition 3. The *probability of a BCQ* Q w.r.t. a PDB $\mathcal{P}$ is

$$\mathrm{P}_{\mathcal{P}}(\mathsf{Q}) = \sum_{\omega \models \mathsf{Q}} \mathrm{P}_{\mathcal{P}}(\omega).$$

For example, considering the PDB in Figure 2 and the query Q_2 defined earlier, $\mathrm{P}_{\mathcal{P}}(\mathsf{Q}_2) = 0.28$.

2.4 The Closed-World Assumption in Practice

Reiter (1978) notes that the CWA presumes a complete knowledge about the domain being represented, and that this assumption is warranted in many cases. For example, when a flight does not appear in an airline database, we can be sure that it never took place. Next, we assess the adequacy of the CWA for probabilistic knowledge bases.

Truncating and Space Blow-up Figure 3 shows the distribution of the probabilities in popular knowledge bases. These automatically constructed PDBs seem hardly probabilistic. Most tuples have a very high probability, placing

PDBs into an almost crisp setting in practice. The underlying reason is that these systems retain only a small fraction of the discovered facts. Facts with a probability below a threshold are discarded, violating the CWA. In fact, PaleoDeepDive contains a wider range of probabilities because it was obtained from the authors before truncation. This mode of operation is not an oversight, but a necessity. It is simply not possible to retain all facts in the Herbrand base. Consider, for instance the `Sibling` relation over a domain of 7 billion people. Storing a single-precision probability for all `Sibling` facts would require 196 exabytes of memory; two orders of magnitude more than the estimated capacity available to Google (Munroe 2015).

Distinguishing Queries Since the CWA is violated in most PDBs, several query answering issues become apparent. Consider for example the queries $\mathsf{Q}_1(\mathsf{pitt}, \mathsf{jolie})$ and Q_2. The former query entails the latter, leading us to expect that $\mathrm{P}(\mathsf{Q}_2) > \mathrm{P}(\mathsf{Q}_1(\mathsf{pitt}, \mathsf{jolie}))$ in an open-word setting (there exist a large number of couples that could satisfy query Q_2). Yet, $\mathrm{P}(\mathsf{Q}_2) = \mathrm{P}(\mathsf{Q}_1(\mathsf{pitt}, \mathsf{jolie})) = 0.28$ in the PDB of Figure 2. For another example, consider the queries $\mathsf{Q}_1(\mathsf{w\_smith}, \mathsf{j\_smith})$ and $\mathsf{Q}_1(\mathsf{thornton}, \mathsf{aniston})$. The former is supported by two facts in the PDB of Figure 2, while the latter is supported by none, which should make it less likely. However, $\mathrm{P}(\mathsf{Q}_1(\mathsf{thornton}, \mathsf{aniston})) = 0$ and $\mathrm{P}(\mathsf{Q}_1(\mathsf{w\_smith}, \mathsf{j\_smith})) = 0$. Taking these observations to the extreme, the query $\mathtt{Inmovie}(x, y) \wedge \neg \mathtt{Inmovie}(x, y)$ is unsatisfiable, yet it evaluates to the same probability as the satisfiable query $\mathsf{Q}_1(\mathsf{thornton}, \mathsf{aniston})$. These counterintuitive results are observed in *real-world data* as well, as we will illustrate after introducing OpenPDBs.

Knowledge-Base Completion and Mining The CWA permeates higher-level tasks that one is usually interested in performing on probabilistic databases. For example, a natural approach to knowledge-base completion learns a probabilistic model from training data. Consider for example a probabilistic rule (Wang, Mazaitis, and Cohen 2013; De Raedt et al. 2015) of the form

$$\mathtt{Costars}(x, y) \xleftarrow{0.8} \mathtt{Inmovie}(x, z), \mathtt{Inmovie}(y, z), \mathtt{Couple}(x, y).$$

To evaluate the quality of this rule for predicting the `Costars` relation, the standard approach would be to quantify the conditional likelihood of the rule based on training data (Sutton and McCallum 2011):

$$D = \{\mathtt{Costars}(\mathsf{w\_smith}, \mathsf{j\_smith}), \mathtt{Costars}(\mathsf{pitt}, \mathsf{jolie})\}$$

The rule predicts $\mathrm{P}(\mathtt{Costars}(\mathsf{w\_smith}, \mathsf{j\_smith})) = 0$, due to the CWA, since one fact is missing from the knowledge base. Hence, the rule gets the likelihood score of zero, regardless of its performance on other tuples in the training data. Another high-level task is to mine frequent patterns in the knowledge base; for example to find the pattern $\mathsf{Q}_1(x, y)$ and report it to the data miner. Again, due to the CWA, the expected frequencies of these patterns will be underestimated (Galárraga et al. 2013).

3 Open-World Probabilistic Databases

Our proposal for open-world probabilistic databases is based on assuming that facts not appearing in a probabilistic database have probabilities in the interval $[0, \lambda]$, for some threshold probability λ. The proposal is formalized using the notion of a *credal set* (Levi 1980), which is a set of probability distributions.

3.1 Syntax, Semantics, and Queries

We start by defining open-world probabilistic databases.

Definition 4. An *open probabilistic database* is a pair $\mathcal{G} = (\mathcal{P}, \lambda)$, where $\mathcal{P}$ is a probabilistic database and $\lambda \in [0, 1]$.

The semantics of an open probabilistic database (OpenPDB) is based on completing probabilistic databases.

Definition 5. A λ-*completion* of probabilistic database $\mathcal{P}$ is another probabilistic database obtained as follows. For each atom t that does not appear in $\mathcal{P}$, we add tuple $\langle t : p \rangle$ to $\mathcal{P}$ for some $p \in [0, \lambda]$.

While a closed probabilistic database induces a unique probability distribution, an open probabilistic database induces a set of probability distributions.

Definition 6. An open probabilistic database $\mathcal{G} = (\mathcal{P}, \lambda)$ induces a set of probability distributions $\mathrm{K}_\mathcal{G}$ such that distribution P belongs to $\mathrm{K}_\mathcal{G}$ iff P is induced by some λ-completion of probabilistic database $\mathcal{P}$.

Note that the set of distributions $\mathrm{K}_\mathcal{G}$ is credal and represents the semantics of OpenPDB $\mathcal{G}$. Intuitively, an OpenPDB represents all possible ways to extend a PDB with new tuples from the open world, with the restriction that the probability of these unknown tuples can never be larger than λ.

Query languages for OpenPDB remain the same as for PDBs, except that queries yield interval-based answers.

Definition 7. The *probability interval of a Boolean query* Q in OpenPDB $\mathcal{G}$ is $\mathrm{K}_\mathcal{G}(\mathsf{Q}) = [\underline{\mathrm{P}}_\mathcal{G}(\mathsf{Q}), \overline{\mathrm{P}}_\mathcal{G}(\mathsf{Q})]$, where

$$\underline{\mathrm{P}}_\mathcal{G}(\mathsf{Q}) = \min_{\mathrm{P} \in \mathrm{K}_\mathcal{G}} \mathrm{P}(\mathsf{Q}) \quad \text{and} \quad \overline{\mathrm{P}}_\mathcal{G}(\mathsf{Q}) = \max_{\mathrm{P} \in \mathrm{K}_\mathcal{G}} \mathrm{P}(\mathsf{Q}).$$

Suppose now that atom t does not appear in OpenPDB $\mathcal{G}$. We then have $\underline{\mathrm{P}}_\mathcal{G}(t) = 0$ and $\overline{\mathrm{P}}_\mathcal{G}(t) = \lambda$. Moreover, if tuple $\langle t : p \rangle$ appears in $\mathcal{G}$, then $\underline{\mathrm{P}}_\mathcal{G}(t) = \overline{\mathrm{P}}_\mathcal{G}(t) = p$. These observations highlight some of the intended semantics of open probabilistic databases.

Reiter (1978) introduced the *open-world assumption* (OWA) as the opposite of the CWA. Under the OWA, a set of tuples no longer corresponds to a single interpretation. Instead, a database corresponds to the set of interpretations that extend it. A similar effect is achieved by OpenPDBs: a set of probabilistic tuples no longer corresponds to a single distribution. Instead, a probabilistic database corresponds to the set of distributions that extend it. In restricting the probabilities of open tuples to lie in $[0, \lambda]$, OpenPDBs follow a rich literature on interval-based probabilities (Halpern 2003), credal networks (Cozman 2000) and default reasoning (Reiter 1980).

3.2 The Open-World Assumption in Practice

We now discuss some implications of the open-world setting before moving to a detailed technical analysis. Consider the queries Q_1 and Q_2 again. We have already noted that $\mathsf{Q}_1(\mathtt{pitt}, \mathtt{jolie})$ entails Q_2, leading us to expect that $\mathrm{P}(\mathsf{Q}_2) > \mathrm{P}(\mathsf{Q}_1(\mathtt{pitt}, \mathtt{jolie}))$ assuming our knowledge is not complete. This is indeed the case for OpenPDBs (for upper probabilities) since there are many worlds with non-zero probability that entail Q_2 but not $\mathsf{Q}_1(\mathtt{pitt}, \mathtt{jolie})$.

We have also observed that an unsatisfiable query is in some cases as likely as a satisfiable one in the closed world. In the open-world setting, the upper probability of a satisfiable query will be greater than the upper probability of an unsatisfiable query. In fact, any unsatisfiable query will still have a zero upper probability in OpenPDBs.

For some further examples, consider the following queries constructed on a portion of the NELL database:

$\mathsf{Q}_3 = \mathtt{Ac}(\mathtt{patt}), \mathtt{Workedfor}(\mathtt{patt}, \mathtt{hwicke}), \mathtt{Di}(\mathtt{hwicke}).$

$\mathsf{Q}_4 = \mathtt{Ac}(x), \mathtt{Inmovie}(x, \mathtt{trainsp}), \mathtt{Mov}(\mathtt{trainsp}), \neg\mathtt{Di}(x).$

$\mathsf{Q}_5 = \mathtt{Ac}(\mathtt{patt}), \mathtt{Workedfor}(\mathtt{patt}, x), \mathtt{Di}(x).$

Here, `Ac` stands for actor, `patt` for Pattinson, `Di` for director, `hwicke` for Hardwicke, `Mov` for movie, and `trainsp` for Trainspotting.

All of the above queries have probability zero on the NELL database, yet we know they correspond to factually true statements. These queries, however, can be distinguished in an open-world setting, as they have varying levels of support. For example, we observe that Q_3 entails Q_5, and posing these queries in the open-world setting, we indeed obtain $\overline{\mathrm{P}}(\mathsf{Q}_5) > \overline{\mathrm{P}}(\mathsf{Q}_3)$ for any non-zero threshold λ. For instance, $\overline{\mathrm{P}}(\mathsf{Q}_5) = 0.82$ and $\overline{\mathrm{P}}(\mathsf{Q}_3) = 0.51$ for $\lambda = 0.3$. Query Q_4 finds actors that starred in the movie Trainspotting and did not direct a movie. Interestingly, there is no world satisfying this query in the NELL database. Evaluating Q_4 in OpenPDBs yields $\overline{\mathrm{P}}(\mathsf{Q}_4) = 0.98$ and $\overline{\mathrm{P}}(\mathsf{Q}_4) = 0.78$ with thresholds 0.7 and 0.3, respectively. These answers are clearly more in line with what one would expect.

The Bayesian learning paradigm is a popular view on machine learning, where the learner maintains beliefs about the world as a probability distribution, and updates these beliefs based on data, to obtain a posterior distribution. In the context of knowledge base completion systems, we observe the following. Given a PDB at time t, such systems gather data D^t to obtain a new model $\mathrm{P}^{t+1}_{\mathcal{P}}(.) = \mathrm{P}^{t}_{\mathcal{P}}(.\,|\,D^t)$. Systems continuously add facts f, that is, set $\mathrm{P}^{t+1}_{\mathcal{P}}(f) > 0$, whereas previously $\mathrm{P}^{t}_{\mathcal{P}}(f) = 0$; an impossible induction for Bayesian learning. This problem is resolved by the open database semantics. Now, facts are not a priori impossible, and adding them does not conflict with the prior beliefs.

4 OpenPDB Query Evaluation

Because OpenPDBs model an infinite set of PDBs, it may seem like an unsurmountable task to efficiently compute intervals $\mathrm{K}_{\mathcal{G}}(\mathsf{Q})$. Fortunately, the problem is simplified by a strong property of credal sets as we employ them here: probability bounds always come from extreme points (Cozman 2000). For OpenPDBs, this means the following.

Definition 8. An *extreme distribution* $\mathrm{P} \in \mathrm{K}_{\mathcal{G}}$ is a distribution where $\mathrm{P}(t) = \overline{\mathrm{P}}_{\mathcal{G}}(t)$ or $\mathrm{P}(t) = \underline{\mathrm{P}}_{\mathcal{G}}(t)$ for all tuples t.

Proposition 9. *For any OpenPDB $\mathcal{G}$ and a query Q, there exist extreme distributions $\mathrm{P}_L, \mathrm{P}_U \in \mathrm{K}_{\mathcal{G}}$ such that $\mathrm{P}_L(\mathsf{Q}) = \underline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q})$, and $\mathrm{P}_U(\mathsf{Q}) = \overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q})$.*

Hence, for OpenPDB query answering, we only need to consider a finite set of distributions, which can be represented by λ-completions where the open-world tuples have an extreme probability.

As for PDBs, general query answering in OpenPDBs is computationally challenging. We discuss two approaches for the case of UCQs.

4.1 Naive Reduction to PDBs

Proposition 9 suggests a naive query answering algorithm: generate all extreme distributions P, compute $\mathrm{P}(\mathsf{Q})$, and report the minimum and maximum. This procedure will terminate, but be exponential in the number of open-world tuples.

For UCQs, however, the monotonicity of the queries allows us to further simplify query evaluation. We can simply choose the minimal (resp. maximal) bound for every tuple. The resulting probability for the UCQ will be minimum (resp. maximum). Thus, the lower probabilities of UCQs w.r.t. OpenPDBs can be computed using a PDB algorithm, by providing it with the set of probabilistic tuples, and ignoring λ. To compute the upper bounds, we can construct a new PDB from the OpenPDB by adding all the open tuples with default upper probabilities λ and simply reuse a standard algorithm developed for PDBs.

Theorem 10. *Given OpenPDB $\mathcal{G} = (\mathcal{P}, \lambda)$ and UCQ Q, consider λ-completion $\mathcal{P}' = \mathcal{P} \cup \{\langle t : \lambda \rangle \mid t \notin \mathcal{P}\}$. Then,*

$$\mathrm{K}_{\mathcal{G}}(\mathsf{Q}) = [\mathrm{P}_{\mathcal{P}}(\mathsf{Q}), \mathrm{P}_{\mathcal{P}'}(\mathsf{Q})].$$

Notice that the construction in Theorem 10 is efficient. It adds all tuples the PDB, which grows polynomially in the domain size. Unfortunately, this is impractical for PDBs with a large domain. Indeed, on the `Sibling` example from Section 2.4, the upper bound would have to be computed on a 196 exabyte closed-world PDB. Thus, an important question is whether this grounding can be avoided. We investigate this in the next section.

4.2 Specialized OpenPDB Algorithm: $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$

We present Algorithm 1 to compute upper bounds of monotone CNF queries on OpenPDBs (lower bounds can be computed with any closed-world PDB algorithm). Monotone CNFs are the dual of UCQ. Hence, this algorithm also computes the probability of UCQ queries.[4] Algorithm 1 is an adaptation of the $\mathbf{Lift}^{\mathbf{R}}$ algorithm presented in Gribkoff, Van den Broeck, and Suciu (2014), which goes back to the algorithm of Dalvi and Suciu (2012). We call our algorithm $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$ where O stands for open. Next, we introduce concepts and notions that are essential to understanding the algorithm. We refer to Gribkoff, Van den Broeck, and Suciu (2014; 2014) for a more detailed description and additional intuitive examples.

[4] Practically, this is achieved by taking the negation of the query and all literals, and returning the complement probability in Step 0.

Algorithm 1 $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}(\mathsf{Q}, \mathcal{P}, \lambda, \mathcal{D})$, abbreviated by $\mathbf{L}(\mathsf{Q}, \mathcal{P})$

Input: CNF Q , prob. tuples $\mathcal{P}$, threshold λ, and domain $\mathcal{D}$.
Output: The upper probability $\overline{\mathrm{P}}_{(\mathcal{P},\lambda)}(\mathsf{Q})$ over domain $\mathcal{D}$.

Step 0 *Base of Recursion*
  if Q is a single ground atom t **then**
    if $\langle t : p\rangle \in \mathcal{P}$ **then return** p **else return** λ
Step 1 *Rewriting of Query*
  Convert Q to union of CNFs: $\mathsf{Q}_{\text{UCNF}} = \mathsf{Q}_1 \vee \ldots \vee \mathsf{Q}_m$
Step 2 *Decomposable Disjunction*
  if $m > 1$ and $\mathsf{Q}_{\text{UCNF}} = \mathsf{Q}_1 \vee \mathsf{Q}_2$ where $\mathsf{Q}_1 \perp \mathsf{Q}_2$ **then**
    $q_1 \leftarrow \mathbf{L}(\mathsf{Q}_1, \mathcal{P}|_{Q_1})$ and $q_2 \leftarrow \mathbf{L}(\mathsf{Q}_2, \mathcal{P}|_{Q_2})$
    return $1 - (1 - q_1) \cdot (1 - q_2)$
Step 3 *Inclusion-Exclusion*
  if $m > 1$ but Q_{UCNF} has no independent Q_i **then**
    return $\sum_{s \subseteq m} (-1)^{|s|+1} \cdot \mathbf{L}(\wedge_{i \in s} \mathsf{Q}_i, \mathcal{P}|_{\wedge_{i \in s} \mathsf{Q}_i})$
Step 4 *Decomposable Conjunction*
  if $\mathsf{Q} = \mathsf{Q}_1 \wedge \mathsf{Q}_2$ where $\mathsf{Q}_1 \perp \mathsf{Q}_2$ **then**
    return $\mathbf{L}(\mathsf{Q}_1, \mathcal{P}|_{\mathsf{Q}_1}) \cdot \mathbf{L}(\mathsf{Q}_2, \mathcal{P}|_{\mathsf{Q}_2})$
Step 5 *Decomposable Universal Quantifier*
  if Q has a *separator variable* x **then**
    let T be all constants as *x-argument* in $\mathcal{P}$
    $q_c \leftarrow \prod_{t \in T} \mathbf{L}(\mathsf{Q}[x/t], \mathcal{P}|_{x=t})$
    $q_o \leftarrow \mathbf{L}(\mathsf{Q}[x/t], \varnothing)$ for some $t \in \mathcal{D} \setminus T$
    return $q_c \cdot q_o^{|\mathcal{D} \setminus T|}$
Step 6 *Fail*

$\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$ assumes that any input query is preprocessed such that (i) it does not contain any constant symbols and (ii) all variables appear in the same order in each predicate occurrence in Q. This preprocessing is efficient.

Steps 1–4 $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$ starts with a *base case* where the query is trivial to evaluate: it is either a probability in $\mathcal{P}$, or the upper bound λ. The *first step* attempts to rewrite Q into a union (disjunction) of CNF sentences. For example, CNF $(R(x) \vee S(y,z)) \wedge (S(x,y) \vee T(x))$ can be rewritten into the union of $(R(x)) \wedge (S(x,y) \vee T(x))$ and $(S(y,z)) \wedge (S(x,y) \vee T(x))$. The *second and third step* apply when the union CNF has multiple disjuncts. They recurse on simplified queries, using standard simplification rules of probability. The second step applies when two sets of CNFs in the union are independent (i.e., share no predicates, denoted $\perp$). Otherwise the third step recurses using the inclusion-exclusion principle. The *fourth step* checks for independent sets of clauses in the CNF and recurses. In the various recursions, the algorithm shrinks the set of tuples $\mathcal{P}$. Specifically, $\mathcal{P}|_Q$ denotes the subset of $\mathcal{P}$ that talks about the predicates that appear in Q.

Step 5 The *fifth step* is the workhorse of $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$, and the key difference with the $\mathbf{Lift}^{\mathbf{R}}$ algorithm of Gribkoff, Van den Broeck, and Suciu (2014). It searches for a special variable, called a *separator*. A separator is a variable that appears in every atom in Q. This means that for any two distinct instantiations t_1, t_2 of the separator, the queries $\mathsf{Q}[x/t_1]$ and $\mathsf{Q}[x/t_2]$ are independent. Hence, by multiplying $\overline{\mathrm{P}}(\mathsf{Q}[x/t])$ for all t in the domain $\mathcal{D}$, we obtain $\overline{\mathrm{P}}(\mathsf{Q})$.

The implementation of step five in $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$ performs one key optimization over this simple multiplication. First, note that x appears in exactly one argument position in Q for every predicate. We call these arguments the x-arguments. Step five partitions the constants in the domain into two sets: (i) the constants T that appear as x-arguments in the tuples in $\mathcal{P}$, and (ii) all other constants, denoted by $\mathcal{D} \setminus T$. For (i), $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$ still enumerates all instantiations of x and computes their probability separately. For (ii), it suffices to compute the probability of a single instantiation of x. All instantiations with constants from $\mathcal{D} \setminus T$ will have the same probability, as they do not depend on the tuples in $\mathcal{P}$. The probability of their conjunction is computed by exponentiation. Moreover, in the recursive calls for $[x/t]$, we can pass along the subset of tuples $\mathcal{P}|_{x=t}$ where all x-arguments are constant t.

Finally, $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$ can fail in *step six*, yielding no answer. We will discuss the meaning of this step in the next section.

5 Data Complexities

Data complexity refers to the complexity of query answering for a fixed query Q, as a function of the set of tuples $\mathcal{P}$. We will study the data complexity for UCQs and UCQ̃s. First, we introduce basic computational complexity notions.

Background Many probabilistic reasoning problems relate to the class PP (Gill 1977); that is the class of languages recognized by a polynomial-time bounded nondeterministic Turing machine that accepts an input iff more than half of the computation paths are accepting. A canonical problem for PP is the majority satisfiability problem (Littman, Majercik, and Pitassi 2001); that is, given a propositional formula ψ, decide whether a majority of the assignments satisfy ψ. PP can be seen as a decision version of the counting class #P (Valiant 1979), for which the canonical problem is #SAT, that is, counting satisfying assignments for a given propositional formula. Beyond PP, the class NP^{PP} is relevant to probabilistic reasoning problems that combine optimization and counting (Park and Darwiche 2004; De Campos and Cozman 2005). These classes relate to standard complexity classes in the following way:

$$\text{Linear} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PP} \subseteq \text{P}^{\text{PP}} \subseteq \text{NP}^{\text{PP}} \subseteq \text{PSpace} \subseteq \text{ExpTime}$$

5.1 Overview

To compare complexities, it will be useful to study decision problems stemming from the computation of $\mathrm{K}_{\mathcal{G}}(\mathsf{Q})$.

Definition 11. Given an OPDB $\mathcal{G}$, query Q and probability p, the *upper probabilistic query evaluation* problem is to decide whether $\overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q}) > p$.

Our key complexity results are illustrated in Figure 4. Briefly, the decision problem of UCQ query evaluation either has linear time data complexity or is PP-complete, depending on the query. This implies a dichotomy, as we will discuss in the next section. Moreover, these complexities are the same for PDBs and OpenPDBs.

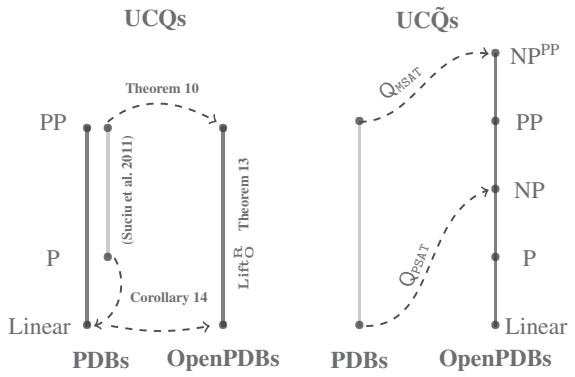

Figure 4: Complexity map for OpenPDBs

For queries with negation, some linear-time PDBs queries can remain linear, and some can become NP-complete on OpenPDBs. Some UCQ̃s that are PP-complete on PDBs can remain PP-complete, and some can become NP^{PP}-complete.

5.2 Monotone Queries: UCQ

The data complexity of query evaluation depends heavily on the structure of the query. In a remarkable result, Dalvi and Suciu (2012) proved a dichotomy: the probability of a UCQ on a PDB can be computed either in PTime or it is #P-hard. As a consequence, the corresponding decision problem is either in PTime or it is PP-complete. The queries that are in PTime are called *safe*. The PP-complete queries are *unsafe*.

The dichotomy of Dalvi and Suciu (2012) is supported by an algorithm similar to $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$. When this algorithm fails, Dalvi and Suciu (2012) prove that the query is PP-complete. When it does not fail, it runs in PTime. This dichotomy-supporting algorithm has one major difference compared to $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$, aside from our support for open-world inference. When it applies the inclusion/exclusion step, it performs cancellations to avoid computing some of the recursive steps. This is a key aspect of the algorithm that ensures efficiency for all PTime queries. We refer to Gribkoff, Van den Broeck, and Suciu (2014) for an in-depth discussion and examples.

For OpenPDBs, it follows from Theorem 10 that the dichotomy for UCQs in PDBs transfers directly.

Corollary 12. *UCQ query evaluation on OpenPDBs is either in PTime, or it is PP-complete. A UCQ is safe in OpenPDBs iff it is safe in PDBs.*

The $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$ algorithm, extended with cancellations in the inclusion-exclusion step, lets us make a stronger statement.

Theorem 13. *Given a UCQ* Q *and OpenPDB* $\mathcal{G}$*,* $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$ *computes* $\overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q})$ *in time* linear *in the size of* $\mathcal{G}$*.*

Moreover, since $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$ fails on all unsafe queries, we obtain a stronger dichotomy.

Corollary 14. *UCQ query evaluation on PDBs or OpenPDBs is either in linear time, or it is PP-complete.*

The linear-time complexity of $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$ is enabled by the shrinking of the database in step five. Briefly, the full proof proceeds as follows. Counting the number of calls in the recursion tree of Algorithm 1, there is a constant number (in the size of $\mathcal{G}$) of calls below each invocation of Line 20, as these calls no longer depend on the data. For the remaining calls to $\mathbf{Lift}^{\mathbf{R}}_{\mathbf{O}}$, we prove that there is a constant number per tuple in $\mathcal{P}$. Steps 0–4 depend only on the query, and not the data. Otherwise, each tuple goes down one path in Step 5. With a constant number of calls per tuple, the entire algorithm can be shown to run in linear time.

Corollary 14 expands the dichotomy of Dalvi and Suciu (2012) from PTime to linear time. Surprisingly, this observation appears to be novel in the PDB literature. Existing linear-time probabilistic query evaluation complexity results, such as the work by Fink and Olteanu (2014), do not apply to the full class of UCQs, nor to the dichotomy-supporting algorithm of Dalvi and Suciu (2012).

5.3 Queries with Negation: UCQ̃s

This section investigates the data complexity of UCQ̃s. As before, we start by showing how probabilistic query evaluation for UCQ̃s w.r.t. OpenPDBs can be decided. Notice that the problem does not trivially reduce to PDBs anymore. The subtlety is that we additionally need to solve an optimization problem. Intuitively, an OpenPDB describes a set of PDBs, each of which differ according to the bounds chosen for the open tuples (cf. Proposition 9). Each choice of bounds yields another PDB, and we are interested in PDBs that yield lower and upper bounds for the query under consideration.

Unsafe Queries We have shown that we can compute the probability of a UCQ in OpenPDBs with a linear operation on the size of the database provided that the corresponding problem in PDBs is linear in the size of the database. This raises a natural question, namely, whether the same holds for UCQ̃s. Unfortunately, very little is known for the computational complexity of UCQ̃s. Consider for instance the UCQ̃ (as its dual CNF) $\mathsf{Q}_{\texttt{MSAT}} := \mathsf{q}_{t_1} \wedge \mathsf{q}_{t_2} \wedge \mathsf{q}_{t_3} \wedge \mathsf{q}_{t_4}$ where:

$$\begin{aligned}
\mathsf{q}_{t_1} &:= \; L(x) \;\vee\; L(y) \;\vee\; L(z) \;\vee\; R_{t_1}(x,y,z),\\
\mathsf{q}_{t_2} &:= \; L(x) \;\vee\; L(y) \;\vee \neg L(z) \;\vee\; R_{t_2}(x,y,z),\\
\mathsf{q}_{t_3} &:= \; L(x) \;\vee \neg L(y) \;\vee \neg L(z) \;\vee\; R_{t_3}(x,y,z),\\
\mathsf{q}_{t_4} &:= \neg L(x) \;\vee \neg L(y) \;\vee \neg L(z) \;\vee\; R_{t_4}(x,y,z).
\end{aligned}$$

$\mathsf{Q}_{\texttt{MSAT}}$ is unsafe in PDBs, and is PP-complete. $\mathsf{Q}_{\texttt{MSAT}}$ is clearly in NP^{PP} in OpenPDBs, by guessing a $\mathrm{P} \in \mathrm{K}_{\mathcal{G}}$ and computing $\mathrm{P}(\mathsf{Q}_{\texttt{MSAT}})$. We show that this is in fact a matching lower bound in OpenPDBs. We do so by providing a reduction from EMAJSAT.

Definition 15 (EMAJSAT)**.** Let ψ be a propositional formula in CNF defined over a (finite) set of propositional variables $V = \{v_1, ..., v_n\}$ and $m \in [1, n]$ an integer. Given the pair (ψ, m), EMAJSAT is the problem of deciding whether there exists an assignment γ of the variables $v_1, ..., v_m$ such that the majority of assignments that extend γ satisfy ψ.

EMAJSAT is NP^{PP}-complete even if we assume that ψ is given in 3-CNF. We construct an OpenPDB $\mathcal{G}$ that encodes ψ. Setting various R_{t_i} tuple probabilities to 0 or 1, we encode the clauses in ψ. Moreover, setting some L-tuple probabilities to 0.5, and leaving others open with $\lambda = 1$, we en-

code which variables are to be maximized over, and which to compute an expectation over.

Using this construction; namely the OpenPDB $\mathcal{G}$ and the query $\mathsf{Q}_{\texttt{MSAT}}$, the full proof shows that it is possible to encode any instance of EMAJSAT. We obtain the following result.

Lemma 16. $\texttt{EMAJSAT}(\psi, m)$ *holds iff* $\overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q}_{\texttt{MSAT}}) > 1/2$.

As a consequence of Lemma 16 and the polynomial construction presented we assert that evaluating $\mathsf{Q}_{\texttt{MSAT}}$ in OpenPDBs is strictly harder than in PDBs, unless PP =NP$^{\text{PP}}$.

Theorem 17. *Upper probabilistic query evaluation for* $\mathsf{Q}_{\texttt{MSAT}}$ *is NP^{PP}-complete w.r.t. OpenPDBs.*

Safe Queries We have shown that unsafe queries can become even harder in OpenPDBs. Does this also apply to safe queries, or do safe queries have certain properties that avoid this hardness? We start inspecting the complexity of safe PDB queries in the context of OpenPDBs. Consider the following UC$\tilde{\text{Q}}$ $\mathsf{Q} = (T(x) \vee \neg F(x,y)) \wedge (F(x,y) \vee \neg L(y))$, represented as CNF. The probability of Q can be computed in PTime w.r.t. a given PDB as shown by Gribkoff, Van den Broeck, and Suciu (2014). This result is non-trivial and it is useful to inspect this query in OpenPDBs. Suppose that we are interested in computing $\overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q})$ for some OpenPDB $\mathcal{G}$. First, observe that we cannot immediately reduce the problem to PDBs as before since Q is non-monotonic. In other words, setting all open tuple probabilities to λ might not yield the maximum probability. On the other hand, we can set the probability of tuples of T to λ and, the tuples of L to 0, in order to maximize the probability of the query. This is sound because the query is monotone in these atoms. As for $F(x,y)$ we have a non-determinism, which can be a potential cause for hardness. Yet, it turns out that this can be avoided too. Simply add the tuple $\langle F(a,b) : \lambda \rangle$ to the PDB if $\overline{\mathrm{P}}_{\mathcal{G}}(T(a)) > \overline{\mathrm{P}}_{\mathcal{G}}(L(y))$ and add $\langle F(a,b) : 0 \rangle$ otherwise.

Intuitively, the structure of this query allows us to locally optimize the choices over the bounds; and hence to avoid non-determinism. What are the characteristics that enabled this local optimization and do these characteristics generalize to all tractable queries?

The answer is, unfortunately, negative. There are queries that are safe for PDBs, but turn out to be NP-complete for OpenPDBs. To show this, we construct a special query $\mathsf{Q}_{\texttt{PSAT}}$ and prove Theorem 18.

Theorem 18. *Probabilistic query evaluation for* $\mathsf{Q}_{\texttt{PSAT}}$ *is in PTime in PDBs. Upper probabilistic query evaluation for* $\mathsf{Q}_{\texttt{PSAT}}$ *is NP-complete in OpenPDBs.*

Theorem 18 is shown using several intermediate results, by reduction from the *3-satisfiability (3SAT)* problem. Briefly, given a propositional formula ψ, SAT is to decide whether ψ is satisfiable. 3SAT is a special case, where ψ is in CNF and every clause contains at most three literals. 3SAT is NP-complete.

The full proof performs the following reduction. First, we encode 3SAT for ψ into the query evaluation problem for query $\mathsf{Q}_{\texttt{MSAT}}$, which is defined earlier, on a carefully constructed OpenPDB.

Second, we obtain the following Lemma

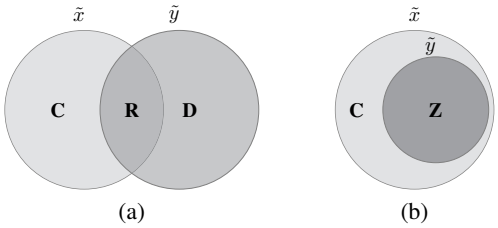

Figure 5: Venn Diag. for q before and after decomposition

Lemma 19. *For any BC$\tilde{\text{Q}}$* q*, there exists an equisatisfiable UC$\tilde{\text{Q}}$* $\mathsf{Q} = \bigwedge_{1 \le i \le n} \mathsf{q}_i$ *such that every* q_i *is individually safe.*

Briefly, Lemma 19 states that any BC$\tilde{\text{Q}}$ can be rewritten into an equisatisfiable UC$\tilde{\text{Q}}$ that is a conjunction of safe queries, only. We transform the query $\mathsf{Q}_{\texttt{MSAT}}$ into an equisatisfiable query $\mathsf{Q}_{\texttt{DSAT}} = \bigwedge_{1 \le i \le n} \mathsf{q}_i$, where every q_i is safe.

Our next result establishes a connection between safe and unsafe query evaluation.

Lemma 20. *For any OpenPDB $\mathcal{G}$ and UC$\tilde{\text{Q}}$* $\mathsf{Q} = \bigwedge_{1 \le i \le n} \mathsf{q}_i$ *where every* q_i *is a safe query, there exists an OpenPDB $\mathcal{G}'$ and UC$\tilde{\text{Q}}$* Q' *such that (i)* $\overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q}) > 0$ *iff* $\overline{\mathrm{P}}_{\mathcal{G}'}(\mathsf{Q}') > C$ *for some constant C, and (ii)* Q' *is a safe query.*

We use this Lemma to transform $\mathsf{Q}_{\texttt{DSAT}}$ into a safe query $\mathsf{Q}_{\texttt{PSAT}}$. By deciding $\overline{\mathrm{P}}_{\mathcal{G}}(\mathsf{Q}_{\texttt{PSAT}}) > C$ on the appropriate, transformed OpenPDB $\mathcal{G}$, we can decide 3SAT for ψ. That completes the proof of Theorem 18, whose details are in the long version of this paper. Next, we give an overview of some technical details for these intermediate results.

Equisatisfiable Rewriting of Queries We introduce some basic notions required for the rewriting procedure, which is in turn used to prove Lemma 19.

Definition 21. Let q be a BC$\tilde{\text{Q}}$. For any variable $x \in \text{Var}(\mathsf{q})$ the set $\{R \mid x \text{ occurs in } R\}$ is the *x-cover ($\tilde{x}$)* of q. Two covers $\tilde{x}$ and $\tilde{y}$ are *pairwise hierarchical* iff $\tilde{x} \cap \tilde{y} \neq \varnothing$ implies $\tilde{x} \subseteq \tilde{y}$ or $\tilde{y} \subseteq \tilde{x}$. A query q is *hierarchical* iff every cover $\tilde{x}, \tilde{y}$ is *pairwise hierarchical*. A *query is of degree n, denoted* $\deg(\mathsf{q})$=n, iff the number of pairs $(\tilde{x}, \tilde{y})$ in $\text{Var}(\mathsf{q})$ that are not pairwise hierachical is n.

Consider now the query $\mathsf{q} = C(x) \vee D(y) \vee R(x,y)$, which is not hierarchical as depicted in Figure 5a using a Venn diagram. In fact, it has the degree 1. It is well-known that hierarchical queries are safe, whereas non-hierarchical ones are unsafe on PDBs (Dalvi and Suciu 2012).

Based on these notions, the long version of this paper defines an algorithm to decompose any BC$\tilde{\text{Q}}$ into a conjunction of queries, each of which is individually safe and computable in PTime. Briefly, the algorithm initializes a set S with the given query. Then, it does the decomposition of a query in $\mathsf{q} \in S$ with $\deg(\mathsf{q}) > 0$ until all queries are safe, i.e. $\deg(S) = 0$. The decomposition splits the query into two parts s and q/s and replaces the part with higher degree (s), with a fresh relation that has the variables in s as arguments. To preserve satisfiability, it adds the logical equivalence $\mathsf{s} \Leftrightarrow Z(x_1, ..., x_n)$ into S, after transforming it into a set

of clauses. Finally, it returns S as the UC$\tilde{\text{Q}}$. Figure 5b shows the effect of adding the formula $Z(x,y) \Leftrightarrow R(x,y) \wedge D(y)$, making all CQs safe.

Creating a Safe Query Evaluating $\mathsf{Q}_{\mathtt{DSAT}}$ is already PP-hard in PDBs, as is $\mathsf{Q}_{\mathtt{MSAT}}$. We are, on the other hand, interested in finding queries that are safe in PDBs, but potentially hard for OpenPDBs. To this purpose, we employ yet another transformation on $\mathsf{Q}_{\mathtt{DSAT}} = \bigwedge_{1 \le i \le n} \mathsf{q}_i$, using Lemma 20. We define $n^2 + n + 1$ clauses as given below:

$$\mathsf{r}_i = \neg H_i \vee \mathsf{q}_i \ , \quad \mathsf{r}_{(i \times n)+j} = \neg H_i \vee \neg H_j$$
$$\mathsf{r}_{n^2+n+1} = H_1 \vee \ldots \vee H_n .$$

where $1 \le i, j \le n$ and the query $\mathsf{Q}_{\mathtt{PSAT}} = \bigwedge_{1 \le i \le n^2+n+1} \mathsf{r}_i$ and prove that $\mathsf{Q}_{\mathtt{PSAT}}$ is NP-complete.

5.4 Domain Complexity

The domain complexity of OpenPDB query evaluation is the complexity for a fixed query and database, as a function of the size of the domain $\mathcal{D}$. Beame et al. (2015) study this complexity (confusingly called data complexity there) in the context of a task called *weighted first-order model counting*. This task is reducible to OpenPDB query evaluation when the database is empty. We refer to Beame et al. (2015) for full details, but note the following.

Corollary 22. *There exists an FO^3 query and set of probabilistic tuples with $\#P_1$-complete domain complexity on OpenPDBs. There exists a BCQ and set of probabilistic tuples with $\#P_1$-hard domain complexity on OpenPDBs.*

Often, however, queries have a PTime domain complexity, in particular queries over two logical variables (Van den Broeck 2013). Some unsafe queries even have PTime domain complexity, for example $\exists x, \exists y, R(x) \wedge S(x,y) \wedge T(y)$.

6 Discussion

Next, we discuss improvements to the OpenPDB semantics and related work.

Restricting the Openness A key challenge is to restrict the open world the provide tighter bounds. We propose specific approaches to exclude spurious possible worlds, and limit the probability mass of open tuples.

One way of restricting the models is to pose constraints on the probability distributions. Such constraints could limit the probability mass of the open world, by for example encoding the average probability of open-world tuples. Notice that if constraints are linear, they can be included into our credal semantics without trouble. Yet, a more fine grained view can be achieved by entropy based approaches. See for instance Lukasiewicz (2000) for an entropy based computation for selecting distributions from a credal set.

Another significant approach is to use an explicit formalism for restricting the models. In fact, it is known that the CWA already causes many subtleties in classical DBs and to address these, ontology based data access (OBDA) has been introduced (Bienvenu et al. 2014). Briefly, OBDA is a means of querying incomplete data sources with the help of a background knowledge provided by an ontology. The most extensively studied formalisms in OBDA are based on Description Logics (Baader et al. 2007). Investigating the computational properties of OBDA in combination with OpenPDBs is left as future work.

Related Work As a credal representation (Levi 1980), OpenPDBs are closely related to credal networks (Cozman 2000). Complexity results for credal networks also show an increase from PTime to NP and PP to NP^{PP} (De Campos and Cozman 2005).

Open-worldness is an important problem in non-probabilistic databases as well (Libkin 2014), and lower or upper bounds on probabilities can be seen as a form of *certain answers* in databases. OWA is common for deterministic knowledge bases, and description logics in particular. OWA is a driving force for these technologies over classical databases in the crisp setting (Patel-Schneider and Horrocks 2006). Analogously, OBDA has been investigated in the probabilistic setting, for both DLs and Datalog (Jung and Lutz 2012; Ceylan and Peñaloza 2015; Gottlob et al. 2013). In Jung and Lutz (2012), authors show how the existing dichotomy in PDBs can be lifted to probabilistic OBDA. However, these formalisms concentrate on computing the minimal probability of an entailment, which corresponds to computing the lower bounds of probabilities, only.

Open information extraction builds knowledge bases with an open-world relational vocabulary (Banko et al. 2007). Moreover, SRL representations do often leave the world open, but there is no tractability, and grounding becomes infeasible at the scale of probabilistic knowledge bases. Modeling the open-world correlations as, for example, a Markov logic network, is a problem in its own right. Such models are not typically part of probabilistic knowledge bases.

To avoid explicitly reasoning about all tuples individually is the topic of *lifted inference* (Poole 2003; Kersting 2012; Van den Broeck 2013). Our work brings together the high-level reasoning of lifted inference and the data-centric reasoning of probabilistic databases.

7 Conclusions

We introduced OpenPDBs, an open-world extension of PDBs. Motivated by probabilistic knowledge bases such as NELL, we provide an efficient algorithm and study the data complexity. For future work, we want to study algorithms for arbitrary queries, based on integer or multi-linear programming (de Campos and Cozman 2007), and strengthen the representation by limiting the effect of the open world.

Acknowledgements We thank Lakshay Rastogi for his help. İ. İlkan Ceylan is supported by the German Research Foundation (DFG) within RoSI (GRK 1907). This research was conducted when he was a visiting student at UCLA.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of databases*, volume 8. Addison-Wesley Reading.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2007. *The Description Logic Handbook:*

Theory, Implementation, and Applications. Cambridge University Press, 2nd edition.

Banko, M.; Cafarella, M. J.; Soderland, S.; Broadhead, M.; and Etzioni, O. 2007. Open information extraction for the web. In *Proc. of IJCAI'07*, volume 7, 2670–2676.

Beame, P.; Van den Broeck, G.; Suciu, D.; and Gribkoff, E. 2015. Symmetric Weighted First-Order Model Counting. In *Proc. of PODS'15*, 313–328. ACM Press.

Bienvenu, M.; Cate, B. T.; Lutz, C.; and Wolter, F. 2014. Ontology-based data access: A study through disjunctive datalog, csp, and mmsnp. *ACM Trans. Database Syst.* 39(4):33:1–33:44.

Bishop, C. M. 2006. *Pattern recognition and machine learning*. Springer.

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning structured embeddings of knowledge bases. In *AAAI'11*.

Ceylan, İ. İ., and Peñaloza, R. 2015. Probabilistic Query Answering in the Bayesian Description Logic BEL. In *Proc. of SUM'15*, volume 9310 of *LNAI*, 21–35. Springer.

Cozman, F. G. 2000. Credal networks. *AIJ* 120(2):199–233.

Dalvi, N., and Suciu, D. 2012. The dichotomy of probabilistic inference for unions of conjunctive queries. *JACM* 59(6):1–87.

De Campos, C. P., and Cozman, F. G. 2005. The inferential complexity of bayesian and credal networks. In *Proc. of IJCAI'05*, AAAI Press, 1313–1318.

de Campos, C. P., and Cozman, F. G. 2007. Inference in credal networks through integer programming. In *Proc. of SIPTA*.

De Raedt, L.; Dries, A.; Thon, I.; Van den Broeck, G.; and Verbeke, M. 2015. Inducing probabilistic relational rules from probabilistic examples. In *Proc. of IJCAI'15*.

Dong, X. L.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K.; Strohmann, T.; Sun, S.; and Zhang, W. 2014. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *Proc. of ACM SIGKDD'14*, KDD'14, 601–610. ACM.

Fader, A.; Soderland, S.; and Etzioni, O. 2011. Identifying relations for open information extraction. In *Proceedings of EMNLP*, 1535–1545. Ass. for Computational Linguistics.

Fink, R., and Olteanu, D. 2014. A dichotomy for non-repeating queries with negation in probabilistic databases. In *Proc. of PODS*, 144–155. ACM.

Fink, R., and Olteanu, D. 2015. Dichotomies for Queries with Negation in Probabilistic Databases. *ACM Transactions on Database Systems (TODS)*.

Galárraga, L. A.; Teflioudi, C.; Hose, K.; and Suchanek, F. 2013. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proc. of WWW'2013*, 413–422.

Gill, J. 1977. Computational complexity of probabilistic turing machines. *SIAM Journal on Computing* 6(4):675–695.

Gottlob, G.; Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2013. Query answering under Probabilistic Uncertainty in Datalog +/- Ontologies. *Ann. Math. AI* 69(1):37–72.

Gribkoff, E.; Suciu, D.; and Van den Broeck, G. 2014. Lifted probabilistic inference: A guide for the database researcher. *Bulletin of the Technical Committee on Data Engineering* 37(3):6–17.

Gribkoff, E.; Van den Broeck, G.; and Suciu, D. 2014. Understanding the Complexity of Lifted Inference and Asymmetric Weighted Model Counting. In *Proc. of UAI'14*, 280–289. AUAI Press.

Halpern, J. Y. 2003. *Reasoning about uncertainty*. MIT Press.

Hinrichs, T., and Genesereth, M. 2006. Herbrand logic. Technical Report LG-2006-02, Stanford University.

Hoffart, J.; Suchanek, F. M.; Berberich, K.; and Weikum, G. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. In *Proc. of IJCAI'2013*, 3161–3165. AAAI Press.

Jung, J. C., and Lutz, C. 2012. Ontology-Based Access to Probabilistic Data with OWL QL. In *Proc. of ISWC'12*, volume 7649 of *LNCS*, 182–197. Springer Verlag.

Kersting, K. 2012. Lifted probabilistic inference. In *Proc. of ECAI'12*, 33–38. IOS Press.

Levi, I. 1980. *The Enterprise of Knowledge*. MIT Press.

Libkin, L. 2014. Certain answers as objects and knowledge. In *Proc. of KR'14*. AAAI Press.

Littman, M. L.; Majercik, S. M.; and Pitassi, T. 2001. Stochastic Boolean Satisability. *J. of Automated Reasoning* 27(3):251–296.

Lukasiewicz, T. 2000. Credal networks under maximum entropy. In *Proc. of UAI'00*, 363–370.

Milch, B.; Marthi, B.; Russell, S.; Sontag, D.; Ong, D. L.; and Kolobov, A. 2007. Blog: Probabilistic models with unknown objects. *Statistical relational learning* 373.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *Proc. of ACL-IJCNLP*, 1003–1011.

Mitchell, T.; Cohen, W.; Hruschka, E.; Talukdar, P.; Betteridge, J.; Carlson, A.; Dalvi, B.; and Gardner, M. 2015. Never-Ending Learning. In *Proc. of AAAI'15*. AAAI Press.

Munroe, R. 2015. Google's datacenters on punch cards.

Park, J. D., and Darwiche, A. 2004. Complexity Results and Approximation Strategies for MAP Explanations. *JAIR* 21(1):101–133.

Patel-Schneider, P. F., and Horrocks, I. 2006. Position paper: a comparison of two modelling paradigms in the semantic web. In *Proc. of WWW'06*, 3–12. ACM.

Poole, D. 2003. First-order probabilistic inference. In *Proc. IJCAI'03*, volume 3, 985–991.

Reiter, R. 1978. On closed world data bases. *Logic and Data Bases* 55–76.

Reiter, R. 1980. A logic for default reasoning. *Artificial intelligence* 13(1):81–132.

Shin, J.; Wu, S.; Wang, F.; De Sa, C.; Zhang, C.; and Ré, C. 2015. Incremental knowledge base construction using deepdive. *Proc. of VLDB* 8(11):1310–1321.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proc. of NIPS'13*, 926–934.

Suciu, D.; Olteanu, D.; Ré, C.; and Koch, C. 2011. Probabilistic Databases.

Sutton, C., and McCallum, A. 2011. An introduction to conditional random fields. *Machine Learning* 4(4):267–373.

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

Van den Broeck, G. 2013. *Lifted Inference and Learning in Statistical Relational Models*. Ph.D. Dissertation, KU Leuven.

Wang, W. Y.; Mazaitis, K.; and Cohen, W. W. 2013. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *Proc. of CIKM*, 2129–2138. ACM.

Wu, W.; Li, H.; Wang, H.; and Zhu, K. Q. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proc. of SIGMOD*, 481–492. ACM.