Query Answering in Probabilistic Data and Knowledge Bases

Dissertation

zur Erlangung des akademischen Grades

Doctor rerum naturalium (Dr. rer. nat.)

vorgelegt an der

Technischen Universität Dresden

Fakultät Informatik

eingereicht von

İsmail İlkan Ceylan, M.Sc.

geboren am 6. Januar 1986 in Tefenni, Türkei

verteidigt am 29.11.2017

Gutachter:	Gutachter:	Fachreferentin:
Prof. Franz Baader	Prof. Georg Gottlob	Prof. Christel Baier
Technische Universität Dresden	University of Oxford	Technische Universität Dresden

Dresden, im April 2018

Abstract

Probabilistic data and knowledge bases are becoming increasingly important in academia and industry. They are continuously extended with new data, powered by modern information extraction tools that associate probabilities with knowledge base facts. The state of the art to store and process such data is founded on probabilistic database systems, which are widely and successfully employed. Beyond all the success stories, however, such systems still lack the fundamental machinery to convey some of the valuable knowledge hidden in them to the end user, which limits their potential applications in practice. In particular, in their classical form, such systems are typically based on strong, unrealistic limitations, such as the *closed-world assumption*, the *closed-domain assumption*, the *tuple-independence assumption*, and the *lack of commonsense knowledge*. These limitations do not only lead to unwanted consequences, but also put such systems on weak footing in important tasks, querying answering being a very central one. In this thesis, we enhance probabilistic data and knowledge bases with more realistic data models, thereby allowing for better means for querying them. Building on the long endeavor of unifying logic and probability, we develop different rigorous semantics for probabilistic data and knowledge bases, analyze their computational properties and identify sources of (in)tractability and design practical scalable query answering algorithms whenever possible. To achieve this, the current work brings together some recent paradigms from logics, probabilistic inference, and database theory.

Acknowledgments

I would first like to express my deepest gratitude to my supervisor Franz Baader for being supportive at all stages of my doctoral studies. Thomas Lukasiewicz, from the University of Oxford, was continuously advising me and I am very much indebted to him. My deepest gratitudes extend to Adnan Darwiche and Guy Van den Broeck, whom I had the chance to work with, during a research visit to University of California, Los Angeles. I am extremely thankful to Rafael Peñaloza, for his guidance during the early phases of my doctoral work, and to Stefan Borgwardt, for his excellent support during the later phases of my doctoral work. I am very thankful to Georg Gottlob, who kindly accepted, on a rather short notice, to act as an external reviewer of this dissertation.

This work was financially supported by the Deutsche Forschungsgemeinschaft (DFG) in the Research Training Group GRK 1907 (RoSI), and afterwards in the Collaborative Research Center 912 (HAEC). It was the exceptional financial framework of RoSI, which made the long-term research visits possible, in the first place. I am pleased to be offered such an opportunity, and am thankful to all who contributed to such a framework. On the administrative side, our chair is surely the luckiest one: Many thanks to Kerstin, our secretary, who was not only dealing with very intricate bureaucratic work, but also pursuing this in the most cheerful way possible.

Special thanks to the colleagues/friends from the Chair for Automata Theory; Stephan Böhme, Andreas Ecke, Oliver Fernández Gil, Patrick Koopmann, Pavlos Marantidis, Maximilian Pensel, Veronika Thost, Anni-Yasmin Turhan and Benjamin Zarrieß; and old members/visitors of the chair; Jieying Chen, Yue Ma, Theofilos Mailis and Julian Mendez. Almost all of them had to continuously deal, and have done so very well, with a person who would walk into their office at random, and even worse, with accompanying random subjects. The colleagues from other chairs were not exempt from this; in particular, David Carral, Irina Dragoste, and Lukas Schweizer were among those, who suffered, and I owe them many thanks for their kindness.

I have always been encouraged, and motivated by my friends Ahmad Ahmadov, Kerem Kaynar, Andrey Rivkin, Umut H. Toprak, İsmail Yüksel, and Murat Zabcı, to write my thesis, and their support was, as always, very much appreciated. Finally, I was thrilled by the continuous support of my parents Anakadın Ceylan and Osman Ceylan; and, of my brother Ramazan Volkan Ceylan. To them, I owe the most: “Her şey için teşekkür ederim; iyi ki varsınız!”

Contents

List of Figures

List of Tables

List of Symbols

First-Order Logic

σ	Relational vocabulary	13
$\mathbf{R}$	Set of predicate names	13
$\mathbf{C}$	Set of constant names	13
$\mathbf{V}$	Set of variable names	13
$\neg, \wedge, \vee$	Logical connectives	14
$\exists, \forall$	Logical quantifiers	14
$\mathcal{T}$	Logical theory	14
Φ, Ψ	Formulas of first-order logic	14
FO	Class of first-order logic formulas	14
$\exists$FO	Existential fragment of first-order logic	14
$\forall$FO	Universal fragment of first-order logic	14
CNF	Class of formulas in conjunctive normal form	14
DNF	Class of formulas in disjunctive normal form	14
$\exists$CNF	Class of formulas in existential conjunctive normal form	14
$\forall$CNF	Class of formulas in universal conjunctive normal form	14
$\exists$DNF	Class of formulas in existential disjunctive normal form	14
$\forall$DNF	Class of formulas in universal disjunctive normal form	14
$\mathcal{I}$	First-order interpretation	15
$\Delta^{\mathcal{I}}$	Interpretation domain	15
ν	Variable assignment	15
$\models$	Entailment relation	15
φ, ψ	Propositional formulas	17

Probability Theory

V	Set of random variables	18
Ω	Sample space	18
$\mathscr{A}$	Event space	18
ω	Possible world	18
P	Probability distribution	19
θ	Event	19

Databases

$\mathcal{D}$	Database	33
UCQ	Unions of Conjunctive Queries	35

Ontologies

Σ	Datalog$^{\pm}$ program (=ontology)	109
$\mathcal{T}$	TBox (=ontology)	112
$\mathcal{A}$	ABox	112
$\mathcal{K}$	Knowledge Base	112
$(Q,\mathcal{T})$	Ontology-mediated query	114
$Q_{\mathcal{T}}$	FO-rewriting of Q with respect to an ontology $\mathcal{T}$	115

Probabilistic Databases

$\mathcal{P}$	Probabilistic Database	37
$\mathrm{P}_{\mathcal{P}}(Q)$	Probability of Q in the PDB $\mathcal{P}$	39
$\mathsf{PQE}(\mathsf{Q})$	Probabilistic Query Evaluation	39
λ	A threshold value	57
$\mathcal{G}$	Open-World Probabilistic Database	57
$\mathrm{K}_{\mathcal{G}}$	(Credal) set of distributions	57
$\underline{\mathrm{P}}_{\mathcal{G}}(Q)$	Lower probability of Q in the OpenPDB $\mathcal{G}$	58
$\overline{\mathrm{P}}_{\mathcal{G}}(Q)$	Upper probability of Q in the OpenPDB $\mathcal{G}$	58
$\underline{\mathsf{P}}\mathsf{QE}(\mathsf{Q})$	Lower Probabilistic Query Evaluation	60
$\overline{\mathsf{P}}\mathsf{QE}(\mathsf{Q})$	Upper Probabilistic Query Evaluation	60
$\mathcal{B}$	Bayesian Network	84
MPE	Most Probable Explanation	85
MAP	Maximum a Posteriori Probability	86
$\mathsf{MPD}(\mathsf{Q})$	Most Probable Database	89
$\mathsf{MPH}(\mathsf{Q})$	Most Probable Hypothesis	96

Probabilistic Knowledge Bases

$\mathrm{P}_{\mathcal{P}}(Q, \Sigma)$	Probability of the OMQ (Q, Σ) in the PDB $\mathcal{P}$	120
$\mathsf{PQE}(\mathsf{Q}, \mathcal{L})$	Probabilistic Query Evaluation for OMQs	121
$\underline{\mathsf{P}}\mathsf{QE}(\mathsf{Q}, \mathcal{L})$	Lower Probabilistic Query Evaluation for OMQs	128
$\overline{\mathsf{P}}\mathsf{QE}(\mathsf{Q}, \mathcal{L})$	Upper Probabilistic Query Evaluation for OMQs	128
$\mathsf{MPD}(\mathsf{Q}, \mathcal{L})$	Most Probable Database for OMQs	136
$\mathsf{MPH}(\mathsf{Q}, \mathcal{L})$	Most Hypothesis Database for OMQs	141

Bayesian Ontology Languages

$\underline{\mathrm{P}}_{\mathcal{K}}(Q)$	Lower probability of Q w.r.t. $\mathcal{K}$	154
$\overline{\mathrm{P}}_{\mathcal{K}}(Q)$	Upper probability of Q w.r.t. $\mathcal{K}$	154
$\underline{\mathsf{B}}\mathsf{QE}(Q, \mathcal{L})$	Lower Bayesian Query Evaluation	154
$\overline{\mathsf{B}}\mathsf{QE}(Q, \mathcal{L})$	Upper Bayesian Query Evaluation	154
$\mathsf{MLC}(Q, \mathcal{L})$	Most Likely Context	166
$\mathsf{MLW}(Q, \mathcal{L})$	Most Likely Context	166
$\mathcal{B}_{\rightarrow}$	Two-slice Bayesian Network	171
$\mathcal{D}$	Dynamic Bayesian Network	171
$\mathrm{P}_{\mathcal{K}}(Q, t)$	Probability of Q w.r.t. $\mathcal{K}$ at time t	176

Part I

Overview and Foundations

Chapter 1

Introduction

1.1 Logic and Probability

Logic is known as the science of *modeling* and *reasoning*. In *mathematical logics*, reasoning is performed over statements, which have a (potential) truth value, such as being *true* or *false*. Such statements are of a certain form and have a precise meaning, where the former refers to the *syntax*, and the latter to the *semantics* of the logical language. Reasoning is then the task of entailing implicit consequences from an explicitly given set of *sentences* using a *sound* procedure.

Despite its deeply theoretical nature, logic has been *unusually effective* in computer science, as noted in (J. Halpern, Harper, Immerman, Kolaitis, Vardi, and Vianu 2001), with applications to artificial intelligence (AI), knowledge representation and reasoning, finite model theory, databases, model checking, computational complexity, program verification, planning and software modeling, among others. Despite its effectiveness, traditionally, logical reasoning takes place in a deterministic universe, where any statement is either *true* or *false*, with no intermediate truth value. However, for a variety of reasons, which we will elaborate later, in artificial intelligence, it is often desirable to have a rigorous framework, in which one can form *uncertain statements*.

Probability theory is the most widely adopted mathematical framework to formally capture the concept of uncertainty. First axiomatized by (Kolmogorov 1933), probability theory is now the universally accepted model of uncertainty. Essentially, probability theory provides the foundations for modeling *probabilistic events;* that is, events, or statements, which can be true or false with some degree of probability. Pearl's development of Bayesian Networks (BNs), a type of probabilistic graphical model, was a key step in AI (Pearl 1988). The intuition behind their success is the capacity of compactly representing a *joint probability distribution* over an event space. Important to note is that advances in probabilistic graphical models have led to the field of *statistical*, or *probabilistic* AI, which separated itself from classical approaches to AI.

The motivation behind marrying logic and probability is to combine the capacity of probability theory to handle pervasive uncertainty with the capacity of mathematical logic to exploit the structure of formal argument. We will take a first glance at some proposals to provide a historical background and discuss some milestones in the area.

Unifying Logical and Probabilistic Approaches

Unifying logic and probability has been a very old endeavor, dating back to scientists such as Leibniz, Boole and Carnap; for a detailed historical background on the subject, we refer the reader to (Hailperin 1984). In the context of AI, perhaps the most ambitious work

was Nilsson's probabilistic logic (Nilsson 1986), which is an extension of propositional logic, in that it allows linear inequalities representing probability bounds for propositions. Nilsson's logic is later supported by an axiomatization in (Fagin, J. Y. Halpern, and Megiddo 1990). It is also extended to incorporate conditional probabilities and a deductive calculus is given in (Frisch and Haddawy 1994). Halpern proposes and expressive first-order probabilistic logic (J. Y. Halpern 1990) and makes a distinction between subjective and statistical interpretation of probabilities. There is a vast literature on combining logic and probability; see especially (J. Y. Halpern 2003) and (Bacchus 1990) and the references therein. Importantly, many of these expressive formalisms turned out to be computationally very demanding, if at all decidable.

Given the undesirable computational properties of expressive models, most of the subsequent approaches were based on probabilistic graphical models, which have a rather simple logical structure. Arguably, one of the most recent milestones in combining logic and probability was the notion of *weighted model counting*. It is with the help of this elegant notion, that logical models (as for traditional AI) and probabilistic graphical models (as for probabilistic AI) are considered to be essentially tightly connected (Chavira and Darwiche 2008). As the name suggests, (propositional) *model counting* is the task of counting the number of satisfying assignments of a given propositional formula, which is a natural extension of satisfiability (SAT) of propositional formulas. *Weighted model counting* additionally incorporates a weight function over the set of all possible assignments. Unsurprisingly, it has been noted that probabilistic inference in Bayesian networks can be reduced to performing *weighted model counting* over weighted propositional theories (Chavira and Darwiche 2008). Thus, weighted model counting, being a purely logical notion, serves as the common ground among many different formalisms. As a consequence, logical and probabilistic models are becoming closer and analogously to SAT solvers (of the automated reasoning community), recent years witness significant effort in building solvers that can do model counting and beyond.

Another major milestone was through the shift from propositional probabilistic models to *relational* probabilistic models in AI. In propositional logic, one is concerned with *propositions* and their interrelationships, which does not provide adequate means for modeling relational domains, which are very common in real life. In other words, as opposed to first-order languages, propositional logic can not express logical *relationships* and *properties* that involve the *parts* of a statement smaller than the elementary statements making it up. As a very simple example, consider the propositions *"tim is a finch"*, *"tim likes seed"* and *"there is a finch that likes seed"*. By simple common sense reasoning, one can exploit that the proposition *"there is a finch that likes seed"* is actually already a consequence of the previous propositions. On the other hand, it is not possible to make such a deduction in propositional logic unless this is explicitly modeled in this way, since logical relationships and properties that involve parts of elementary propositions such as their subjects and predicates are not taken into consideration. The expressive power of BNs and similar models is essentially propositional. As Russell points out, the world is *uncertain* and it has *things* in it (Russell 2015), which identifies the need for the combination of probabilistic models with first-order representations. Thus, recent work focuses on statistical and relational AI (Getoor and Taskar 2007; Raedt, Kersting, and Natarajan 2016), which are *relational* representations of probabilistic graphical models. Statistical relational models build on ideas from probability theory and logic while

incorporating tools from databases and programming languages to represent structure. These models typically employ simplifying assumptions on the logical language as well as on the probabilistic model to keep the resulting formalism simple and can thus also be viewed as modest versions of more elaborate models such as (J. Y. Halpern 2003) and (Bacchus 1990).

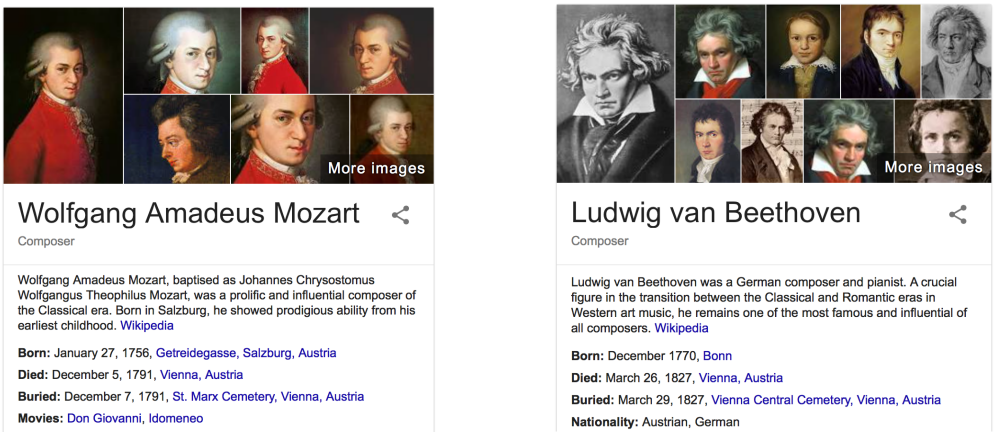
We take these desiderata further in taking into consideration the shift to Big Data oriented models. Unfortunately, most of the simplifying assumptions underlying common probabilistic data models turn out to be inadequate for many application domains. The striving research question is: Is it possible to develop more *realistic data models* to marry logic and probability in the context of probabilistic Big Data? This question brings us to the core ideas underlying the thesis.

1.2 Probabilistic Data and Knowledge Bases

In recent years, there has been a strong interest in building *large-scale probabilistic knowledge bases* from data in an automated way, which has resulted in a number of systems, such as DeepDive (Shin et al. 2015), NELL (Mitchell et al. 2015), Reverb (Fader, Soderland, and Etzioni 2011), Yago (Hoffart, Suchanek, Berberich, and Weikum 2013), Microsoft's Probase (Wu, Li, Wang, and Zhu 2012), IBM's Watson (D. A. Ferrucci 2012), and Google's Knowledge Vault (Dong et al. 2014). These systems continuously crawl the Web and extract *structured information*, and thus populate their databases with millions of entities and billions of tuples. The research centered around large-scale knowledge bases serves as a new era in unifying logic and probability.

To what extent can these search and extraction systems help with real-world use cases? This turns out to be an open-ended question. While being at their infancy, systems such as DeepDive are now routinely used to build knowledge bases for domains such as *paleontology*, *geology*, *medical genetics*, and *human movement*; see e.g. (Ku, Hicks, Hastie, Leskovec, Ré, and Delp 2015) and (Peters, Zhang, Livny, and Ré 2014); it also serves as an important tool for the fight against human trafficking (Greenemeier 2015). IBM's Watson is revolutionizing *health-care systems* (D. Ferrucci, Levas, Bagchi, Gondek, and Mueller 2013) and many other application domains of *life sciences*. Google's Knowledge Vault has compiled more than a billion facts from the Web and is primarily used to improve the quality of search results on the Web. Currently, it can even estimate the trustworthiness of more than 119M sources (Dong et al.) using the extensive knowledge stored (Marsden 2015).

From a broader perspective, the quest for building large knowledge bases serves as a new dawn for AI research. Fields such as *information extraction*, *natural language processing (e.g., question answering)*, *relational and deep learning*, *knowledge representation and reasoning*, and *databases* are taking initiative towards a common goal. Querying large-scale probabilistic knowledge bases is commonly regarded to be at the heart of these efforts. Perhaps the most visible application of these probabilistic knowledge bases is in search engines. Nowadays, the standard list of relevant web pages is often augmented with a table of structured data that pertains to the search query. For instance, the search for Mozart (Figure 1.1a) or Beethoven (Figure 1.1b) shows a box identifying



(a) Mozart (b) Beethoven

Figure 1.1: Information boxes for the Google search (a) Mozart and (b) Beethoven. The information boxes are linked to the underlying knowledge base that provides basic information about the search entities.

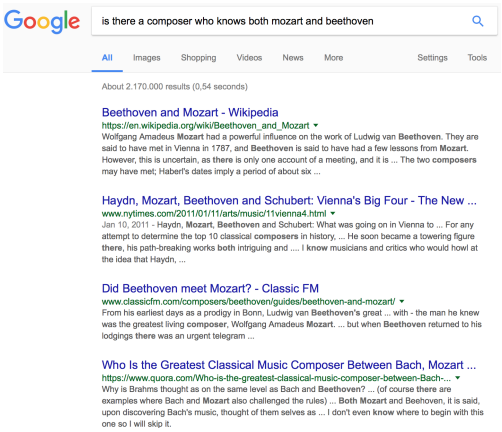
their children, spouse, students, place of birth, et cetera, which is clearly linked to the underlying knowledge base.

Beyond all the success stories, however, probabilistic knowledge bases still lack the fundamental machinery to convey some of the valuable knowledge hidden in them to the end user (Weikum, Hoffart, and Suchanek 2016), which seriously limits their potential applications in practice. For instance, the information shown next to search results in Google search is highly curated, only showing facts that the search engine provider is absolutely certain about. Other, more uncertain information is hidden from the user. Moreover, there is no support for asking higher-level queries on top of these databases. Even asking for a simple join query, "is there a composer who knows both Mozart and Beethoven", is currently beyond the scope of these systems (Dong, Gabrilovich, Heitz, Horn, Lao, Murphy, Strohmann, Sun, and Zhang 2014), despite the fact that a potential answer, Haydn, is an entity known by the probabilistic knowledge base (Figure 1.2a). What makes this simple join query so hard to process and why can this knowledge not be transmitted to the user? These questions constitute our global motivation and the answers are linked to deep theoretical problems as well as to technical limitations, as we outline next.

1.2.1 Probabilistic Databases

Most importantly, many large-scale knowledge bases are essentially 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, Bills, Snow, and Jurafsky 2009). For knowledge-base completion – the task of deriving new facts from existing knowledge – statistical relational learning algorithms make use of embeddings (Bordes, Weston, Collobert, and Bengio 2011; Socher, Chen, Manning, and Ng 2013) or probabilistic rules (De Raedt, Dries, Thon, Van den

(a) Haydn



(b) Is there a composer who knows Mozart and Beethoven?

Figure 1.2: Inability to perform high-level queries on large knowledge base systems. Even though Mozart (Figure 1.1a), Beethoven (Figure 1.1b), and Haydn (Figure 1.2a) are all entities known by the probabilistic knowledge base, the question whether there is a composer who knows both Mozart and Beethoven (Figure 1.2b) cannot be answered by this system.

Broeck, and Verbeke 2015; Wang, Mazaitis, and Cohen 2013). In both settings, the output is a predicted fact with its probability. It is therefore common to interpret such knowledge through a probabilistic semantics.

The most basic model is that of *tuple-independent probabilistic databases* (PDBs) (Suciu, Olteanu, Ré, and Koch 2011), which indeed underlies many of these systems (Dong, Gabrilovich, Heitz, et al. 2014; Shin, Wu, Wang, De Sa, Zhang, and Ré 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).

Probabilistic databases typically lack a suitable handling of incompleteness in practice. In particular, each of the above systems encodes only a portion of the real-world, and this description is necessarily *incomplete*. For example, according to YAGO, the average number of children per person is 0.02 (Galárraga, Razniewski, Amarilli, and Suchanek 2017). However, when it comes to querying, most of these systems employ the CWA, i.e., any fact that is not present in the knowledge base is assigned the probability 0, and thus assumed to be impossible, although it actually has some unknown probability in $[0, 1]$. As a closely related problem, it is common practice to view every extracted fact as an *independent* Bernoulli variable, i.e., any two facts are probabilistically independent.

Example 1.1 We illustrate the above serious deficiencies on the following very simple tuple-independent PDB, which assigns the probability 0.5 to several self-explaining facts:

$$\langle \mathsf{Composer}(\mathsf{haydn}) : 0.5 \rangle , \langle \mathsf{TeacherOf}(\mathsf{haydn}, \mathsf{beethoven}) : 0.5 \rangle ,$$
$$\langle \mathsf{Knows}(\mathsf{haydn}, \mathsf{beethoven}) : 0.5 \rangle , \langle \mathsf{FriendOf}(\mathsf{haydn}, \mathsf{mozart}) : 0.5 \rangle .$$

Under the CWA, all the missing facts have the probability 0, i.e., they are false. Consequently, the following two queries both evaluate to the probability **0** over the PDB:

$$Q_1 \coloneqq \exists x\,(\mathsf{TeacherOf}(x, \mathsf{beethoven}) \wedge \mathsf{BornIn}(x, \mathsf{austria})),$$
$$Q_2 \coloneqq \exists x\,(\mathsf{Person}(x) \wedge \neg\mathsf{Person}(x)).$$

In particular, the fact $\mathsf{BornIn}(\mathsf{haydn}, \mathsf{austria})$ is assumed to have the probability 0 (i.e., to be false); however, this assumption may be incorrect. Indeed, $\mathsf{BornIn}(\mathsf{haydn}, \mathsf{austria})$ may even have the probability 1 (i.e., may be true), which would result in Q_1 having the probability **0.5**.

On the other hand, Q_2 is unsatisfiable and should always have the probability **0**, no matter how incomplete the PDB is. That is, the CWA forces a very flat representation, which makes it impossible to even distinguish a satisfiable query from an unsatisfiable one. Furthermore, under tuple-independence, the query

$$Q_3 \coloneqq \exists x\,(\mathsf{TeacherOf}(x, \mathsf{beethoven}) \wedge \mathsf{Knows}(x, \mathsf{beethoven}))$$

evaluates to the probability $0.5 \cdot 0.5 = \mathbf{0.25}$. But Haydn being a teacher of Beethoven already implies him knowing Beethoven; so, the probability for Q_3 should actually be **0.5**, instead. ◊

These observations become more dramatic once combined with another limitation of these systems; namely, the lack of *commonsense knowledge*, which brings us to ontological knowledge bases.

1.2.2 Ontological Knowledge and Probabilistic Knowledge Bases

The lack of commonsense knowledge is one of the main reasons why some obvious answers can not be retrieved from the PDBs. This shows up in real-world applications of PDBs: especially, in the context of Web search, where the structured information results are linked to the underlying knowledge base.

Example 1.2 A simple join query asking whether there is a composer who knows both Mozart and Beethoven,

$$Q_4 \coloneqq \exists x\,(\mathsf{Composer}(x) \wedge \mathsf{Knows}(x, \mathsf{beethoven}) \wedge \mathsf{Knows}(x, \mathsf{mozart})),$$

is evaluated to the probability 0 and thus cannot be evaluated correctly by these systems (Dong, Gabrilovich, Heitz, et al. 2014). The answer to this query is actually in the knowledge base: it is known that Haydn is (i) a composer, (ii) a friend of Mozart, and (iii) one of the teachers of Beethoven. ◊

In fact, both queries "friend of Mozart", and "teacher of Beethoven", retrieve the correct information pointing to Haydn, a composer known by the knowledge base. However, explicit information about $\mathsf{Knows}(\mathsf{haydn}, \mathsf{mozart})$ is missing, and hence this fact is evaluated to the probability 0 by the CWA.

It is hard to process this simple join query, because current PDBs lack commonsense knowledge: Human beings know that two persons who are friends know each other. Reasoning exploits such basic knowledge to deduce implicit consequences from data,

and this kind of knowledge is essential for querying large-scale PDBs in an uncontrolled environment, i.e., the internet. Therefore, incorporating commonsense knowledge is very important, which is inherently connected to giving up the above completeness assumptions of standard PDBs (CWA and probabilistic independence of facts).

The crucial need to relax the independence assumption has already been recognized in several recent approaches, which are based on Markov logic networks (MLNs) (Gribkoff and Suciu 2016a). All these approaches have in common that they also allow for modeling logical knowledge. However, since they are based on *grounding* probabilistically independent factors of the MLN with known constants, they essentially only allow for encoding *propositional* logical knowledge, and not fully fledged first-order knowledge, as it already occurs in (rather restricted) ontology languages that are used to formulate commonsense knowledge. That is, MLNs cannot express unknown individuals (also called *nulls*), as illustrated by the following example.

Example 1.3 Consider the constraint *"everyone has a parent"*. Unless the parent of an individual a is explicitly mentioned in the PDB, in an MLN this constraint means that a must be assigned a *"parent"* from the known individuals; that is, a completely random person mentioned in the database. This is due to the semantics of MLNs and databases, which employ the closed-domain assumption. Even more dramatically, as the number of objects in the database is fixed, this semantics forces the *parent of* relation to be cyclic, i.e., some individuals will be marked as both *ancestor* and *descendant* of some other individuals, which is clearly absurd. ◊

The *closed-domain assumption*, the assumption that fixes the domain of discourse to a *finite* set of *known* constants, is incorporated as a *de facto standard* in almost all statistical relational models; well-known examples are MLNs (Richardson and Domingos 2006), relational Bayesian networks (Jaeger 1997), and function-free variants of probabilistic logic programs such as ProbLog (De Raedt, Kimmig, and Toivonen 2007). Unfortunately, operating over a closed domain has strong consequences as illustrated in the example.

Ontologies, on the other side, are first-order theories that formalize domain-specific knowledge, thereby allowing for automated reasoning. The most prominent ontology languages in the literature are based on Datalog$^{\pm}$ (Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012; Calì, Gottlob, and Pieris 2012), also studied under the name of *existential rules*, and on Description Logics (DLs) (Baader, Calvanese, McGuinness, Nardi, and Patel-Schneider 2007). Interpreting databases under commonsense knowledge in the form of ontologies is closely related to ontology-based data access (OBDA) (Poggi, Lembo, Calvanese, De Giacomo, Lenzerini, and 2008), which has been widely studied in the context of classical databases, and also addresses the need for open-world querying. In an OBDA scenario, a database query is mediated by a logical interface to make implicit commonsense knowledge explicit: allowing for open-world querying, this results in more complete set of answers for the query.

In this dissertation, we enhance large-scale PDBs with more realistic data models, thereby allowing for better means for querying. We develop different rigorous semantics for probabilistic data and knowledge bases, analyse their computational properties and identify sources of in/tractability and design practical scalable query answering algorithms whenever possible. To achieve this, the current work brings together some

recent paradigms from database theory and logic for probabilistic query evaluation and related inference tasks.

1.3 Structure of the Thesis

The thesis is organized in *four parts*. Part I gives an overview of the work, provides the motivational desiderata and settles the foundational background. Part II focuses on probabilistic databases and on common database query languages. Part III combines probabilistic databases with logical theories, which we refer to as probabilistic knowledge bases. Part IV includes conclusions and an outlook.

Part I: Overview and Foundations. This part is dedicated to provide a general motivation of the work and to give an overview of the foundations of the respective fields that are relevant to the thesis.

Chapter 1. This chapter provides a general introduction and illustrates the main observations that motivated this work. We start with a rather high-level motivation, which is followed by more concrete desiderata illustrated on simple, concrete examples.

Chapter 2. This chapter introduces the main notions used in the thesis. The first section gives an overview on first-order logic. Subsequently, a small introduction to probability theory is given. This chapter also contains basics of the theory of computation and a short introduction to computational complexity theory. We finally note that this part also settles some notation used in this work, which is used throughout the thesis.

Part II: Probabilistic Databases. This part presents the results on Probabilistic Databases and is organized in three chapters. This part also serves as the basis for the results presented in Part III.

Chapter 3. This chapter provides an introduction to databases from a model-theoretic perspective, and contains a review of known results from database theory as well as an overview of existing results on probabilistic query evaluation in PDBs. The connection between probabilistic query evaluation and weighted model counting is also summarized. This chapter also contains the first results presented in the thesis which mainly extend from the results from (Ceylan, Darwiche, and Van den Broeck 2016) and from (Borgwardt, Ceylan, and Lukasiewicz 2017). At the end, this chapter contains an extensive review of the state of the art in PDBs. Importantly, since this work covers a wide spectrum ranging from PDBs to much more sophisticated models including ontologies, some of the related work is deferred to the relevant chapters to allow for a more incremental presentation.

Chapter 4. This chapter introduces *open-world probabilistic databases*, which is proposed as a new data model for probabilistic databases. The main difference between PDBs and open-world probabilistic databases is that the latter does not employ the CWA. We provide a deep discussion of the semantic differences between these models and

compare them with respect to the goals identified in this work. Apart from the semantic results, this chapter also contains a thorough complexity analysis with respect to a wide range of database queries. This analysis includes a data complexity dichotomy result and an efficient lifted inference algorithm. The main results presented here have been previously published in (Ceylan, Darwiche, and Van den Broeck 2016), which was awarded the *Marco Cadoli Best Student Paper Prize* in the 15th International Conference on Principles of Knowledge Representation and Reasoning, 2016. An abridged version of this paper has also appeared as an invited publication in (Ceylan, Darwiche, and Van den Broeck 2017). We significantly extend these results with a detailed treatment of different query languages as well as with new results that go beyond the data complexity.

Chapter 5. In this chapter, two alternative inference tasks for probabilistic databases; namely finding the *most probable database* and the *most probable hypothesis* for a given query are studied. Both of these problems are inspired by the *maximal posterior probability* computations of Probabilistic Graphical Models. These inference tasks can be helpful to exploit the full potential of probabilistic databases. A very detailed complexity analysis is provided for both problems. Most of the results presented in this chapter are based on the previous publication (Ceylan, Borgwardt, and Lukasiewicz 2017). In this work, we additionally close some open problems identified in earlier work, while also paving the way for new open problems.

Part III: Logic and Probabilistic Knowledge Bases. This part extends the results from Part II to also incorporate commonsense knowledge in the form of ontologies. In this setting, database queries are additionally equipped with the expressive power of the ontologies. Following a common naming convention in the field, we refer to such queries as *ontology-mediated queries*. The presentation is organized in three chapters.

Chapter 6. This chapter is dedicated to provide an overview of the prominent ontology languages in the literature. More precisely, we cover Description Logics and Datalog$^{\pm}$ programs, each of which represents a family of knowledge representation formalisms. In particular, we focus on the paradigm of ontology-mediated query answering in these formalisms and recall the computational complexity of this task in the respective languages. This chapter concludes with a discussion regarding the connection between ontology-mediated query answering and (probabilistic) databases.

Chapter 7. In this chapter, the problems introduced in Part II are extended to the case of ontology-mediated queries based on Datalog$^{\pm}$. As a result, this chapter is organized in three main sections in correspondence with Chapters 3, 4 and 5, respectively. First, we extend our results from Chapter 3 and study probabilistic ontology-mediated query evaluation for PDBs. Afterwards, we study probabilistic ontology-mediated query evaluation for open-world probabilistic databases; thus, extend our results from Chapter 4. The study in these sections builds on the previous publication (Borgwardt, Ceylan, and Lukasiewicz 2017) and is also related to (Ceylan, Lukasiewicz, and Peñaloza 2016). Finally, we revisit the most probable database and most probable hypothesis problems from Chapter 5 in the context of ontology-mediated queries based on earlier work (Ceylan,

Borgwardt, and Lukasiewicz 2017). In each of these sections, we provide a thorough complexity analysis. This chapter concludes with an overview of the related work and possible directions for future research.

Chapter 8. In this chapter, Bayesian ontology languages are investigated. These languages extend classical Description Logics with the capability of representing and reasoning over uncertain domains. First, we study ontology-mediated query evaluation in Bayesian ontology languages together with two other reasoning problems, called *most likely context* and *most likely world*. Afterwards, we propose a novel monitoring approach that combines the power of ontology languages with *dynamic* BNs. The resulting formalism is then called dynamic Bayesian ontology languages and allows to make projections about the future states of a system. Bayesian ontology language were first proposed in (Ceylan and Peñaloza 2014b) (see also the related workshop paper (Ceylan and Peñaloza 2014a)), and further studied in (Ceylan and Peñaloza 2014c); afterwards, combined in a journal a paper (Ceylan and Peñaloza 2017) as part of a special issue. This thesis includes a rather minor part of these early results as they are based on standard reasoning tasks (and some of these results were obtained earlier than the thesis work). Our focus is on query answering and our results significantly extend on early work previously presented in (Ceylan and Peñaloza 2015b) by a treatment of arbitrary ontology languages, and many more details. A proof-of-concept implementation for reasoning in Bayesian ontology languages was described in (Ceylan, Mendez, and Peñaloza 2015). Dynamic Bayesian ontology languages were first proposed in (Ceylan and Peñaloza 2015a). There are less closely related publications, which are excluded from the thesis, such as (Ceylan, Lukasiewicz, Peñaloza, and Tifrea-Marciuska 2017).

Part IV: Conclusions. This part consists of only Chapter 9, which summarizes the current work and provides possible directions for future work.

Remark. I served as the main author in the publications listed above and I was always supported by exceptional co-authors, namely Stefan Borgwardt, Adnan Darwiche, Thomas Lukasiewicz, Rafael Peñaloza, and Guy Van den Broeck. The work (Ceylan, Darwiche, and Van den Broeck 2016) is an outcome of a research visit to University of California, Los Angeles (Automated Reasoning Lab, led by Prof. Adnan Darwiche). I also did benefit from the oppurtunity of conducting several short-term visits to University of Oxford, which resulted in a long-term collaboration with Prof. Thomas Lukasiewicz.

Chapter 2

Foundations

This chapter introduces some of the main notions used in this thesis. In order to allow for a more incremental presentation, some of the preliminaries are deferred to later chapters as they only become relevant after those chapters. The first section gives an overview on first-order logic. Subsequently, a small introduction to probability theory is given to an extend that is relevant for the current work. At the end, this chapter contains the very basics of the theory of computation and a short introduction to computational complexity theory.

2.1 First-Order Logic

Logic is known as the science of *reasoning*. In *mathematical logics*, reasoning is performed over *declarative sentences* (or statements), which have a (potential) truth value, such as being *true* or *false*. Such sentences are of a certain form and have a precise meaning, where the former refers to the *syntax*, and the latter to the *semantics* of the logical language. Reasoning is then the task of entailing implicit consequences from an explicitly given set of *sentences* using a *sound* procedure.

There are a plethora of logical languages with different types of reasoning tasks in the literature and we assume familiarity with propositional and quantified Boolean logic. We only recall *first-order logic* as most of the languages that happen to appear in this work, are essentially *fragments* of first-order logic. We only give a brief overview, whereby we also settle some basic notation; for a broader introduction to mathematical logic, we refer to standard texts from the literature (Ebbinghaus, Flum, and Thomas 1994; Enderton 2001; Fitting 1996; Van Dalen 2004).

We focus on the *function-free* fragment of *first-order logic (FOL)*. First, we introduce some basic notions (syntax and semantics of FOL); then, we recall some of the classical results (i.e., soundness and completeness), and finally we discuss some restricting assumptions (i.e., finite, fixed domain assumptions) of first-order logic, all of which are crucial ingredients of this work.

Definition 2.1 (vocabulary, terms, atoms) A *relational vocabulary*, denoted by σ, consists of sets $\mathbf{R}$ of *predicate*, $\mathbf{C}$ of *constant*, and $\mathbf{V}$ of *variable (names)*. The function $\mathsf{ar} : \mathbf{R} \mapsto \mathbb{N}$ associates a natural number to each predicate $\mathsf{P} \in \mathbf{R}$ that defines the (unique) arity of P. A *term* is either a constant or a variable. An *atom* is of the form $\mathsf{P}(s_1, \ldots, s_n)$, where P is an n-ary predicate, and $s_1, \ldots, s_n$ are terms. A *ground atom* is an atom without variables. $\Diamond$

For ease of presentation and to avoid ambiguity (in operator precedence), it is customary to use some auxiliary symbols such as ",", "(", and ")" as part of the vocabulary. We

will make use of such symbols to increase readability without further notice. We will often not mention the vocabulary explicitly if it is clear from the context.

Definition 2.2 (logical connectives, quantifiers) The *logical connectives* are *negation* ($\neg$), *conjunction* ($\wedge$) and *disjunction* ($\vee$). The *logical quantifiers* in FOL are *existential quantifier* ($\exists$) and *universal quantifier* ($\forall$). $\Diamond$

First-order formulas are inductively built from atomic formulas, i.e., atoms, using the logical constructors and quantifiers, as defined next.

Definition 2.3 (formula, sentence) Given a relational vocabulary σ, a *first-order formula* is defined by the syntax rule

$$\Phi \coloneqq \mathsf{P}(s_1, \ldots, s_n) \mid \neg\Phi \mid \Phi \wedge \Phi \mid \exists x.\Phi,$$

where P is an n-ary predicate, $s_1, \ldots, s_n$ are terms, and x is a variable. A *literal* is either an atom or its negation. A *quantifier-free* formula is a formula that does not use a quantifier. A variable x in a formula Φ is *quantified*, or *bound* if it is in the scope of a quantifier; otherwise, it is *free*. A *(first-order) sentence* is a first-order formula without any free variables, also called a *closed formula*, or *Boolean formula*. A *(first-order) theory* is a set of first-order sentences. $\Diamond$

We use the standard shorthand notation and write $\Phi \vee \Psi$ in place of $\neg(\neg\Phi \wedge \neg\Psi)$; $\Phi \rightarrow \Psi$ in place of $\neg\Phi \vee \Psi$; $\Phi \leftrightarrow \Psi$ in place of $(\Phi \rightarrow \Psi) \wedge (\Psi \rightarrow \Phi)$. We also abbreviate the existentially quantified formulas of the form $\neg\exists x.\neg\Phi$ with $\forall x.\Phi$. Finally, the truth symbols $\top$ and $\bot$ are used to abbreviate formulas of the form $\Phi \vee \neg\Phi$, and $\Phi \wedge \neg\Phi$, respectively.

We use a vector notation to denote a sequence of variables $x_1, \ldots, x_n$ by $\vec{x}$. The set of free variables in Φ is denoted by $\mathsf{FV}(\Phi)$, and we write $\Phi(\vec{x})$ where $\mathsf{FV}(\Phi) = \{\vec{x}\}$. It is often very useful to consider *syntactic fragments* of first-order logic formulas, as we define next.

Definition 2.4 (FO-fragments) Let FO be the class of first-order formulas. The class of *existential first-order formulas* ($\exists\mathsf{FO}$) consists of first-order formulas of the form $\exists\vec{x}.\Phi(\vec{x})$, where Φ is any Boolean combination of atoms. The class of *universal first-order formulas* ($\forall\mathsf{FO}$) consists of first-order formulas of the form $\forall\vec{x}.\Phi(\vec{x})$, where Φ is any Boolean combination of atoms.

A *disjunctive clause* is a finite disjunction of literals. A *conjunctive clause* is a finite conjunction of literals. The class of formulas in *existential conjunctive normal form* ($\exists\mathsf{CNF}$) consists of first-order formulas of the form $\exists\vec{x}.\Phi(\vec{x})$; the class of formulas in *universal conjunctive normal form* ($\forall\mathsf{CNF}$) consists of first-order formulas of the form $\forall\vec{x}.\Phi(\vec{x})$, where Φ is a conjunction of disjunctive clauses.

The class of formulas in *existential disjunctive normal form* ($\exists\mathsf{DNF}$) consists of formulas of the form $\exists\vec{x}.\Phi(\vec{x})$; the class of formulas in *universal disjunctive normal form* ($\forall\mathsf{DNF}$) consists of formulas of the form $\forall\vec{x}.\Phi(\vec{x})$, where Φ is a disjunction of conjunctive clauses.

The class of formulas in *conjunctive normal form* (CNF) consists of $\exists\mathsf{CNF}$ and $\forall\mathsf{CNF}$ formulas. The class of formulas in *disjunctive normal form* (DNF) consists of $\exists\mathsf{DNF}$ and $\forall\mathsf{DNF}$ formulas. A formula is *positive* if it contains only positive literals. We also write $k\mathsf{CNF}$, or $k\mathsf{DNF}$ to denote the class of formulas, where k denotes the maximal number of atoms that a clause can contain. $\Diamond$

So far, we purely focused on the syntax and have not yet associated any meaning to these syntactic entities. We shall set up a correspondence between a first-order formula and its meaning in the outside world. The semantics of FOL is given by first-order interpretations, which we define next.

Definition 2.5 (semantics) Given a relational vocabulary σ, a first-order *interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a *non-empty domain*, and $\cdot^{\mathcal{I}}$ is an *interpretation function* that maps every constant name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ of the domain, and every predicate name P with arity n to a subset $\mathsf{P}^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$ of the domain. A *variable assignment* is a function $\nu : \mathbf{V} \mapsto \Delta^{\mathcal{I}}$ that maps variables to domain elements. Given an element $e \in \Delta^{\mathcal{I}}$ and a variable $x \in \mathbf{V}$, we write $\nu[x \mapsto e]$ to denote the variable assignment that maps x to e, and that agrees with ν on all other variables. For an interpretation $\mathcal{I}$ and a variable assignment ν, we define

- $a^{\mathcal{I},\nu} = a^{\mathcal{I}}$ for all constant names $a \in \mathbf{C}$,
- $x^{\mathcal{I},\nu} = \nu(x)$ for all variable names $x \in \mathbf{V}$,
- $\mathsf{P}^{\mathcal{I},\nu} = \mathsf{P}^{\mathcal{I}}$ for all predicate names $\mathsf{P} \in \mathbf{R}$.

Given an interpretation $\mathcal{I}$ and a variable assignment ν, the satisfaction relation ($\models$) is inductively defined as

- $\mathcal{I}, \nu \models \mathsf{P}(s_1, \ldots, s_n)$ if $(s_1^{\mathcal{I},\nu}, \ldots, s_n^{\mathcal{I},\nu}) \in \mathsf{P}^{\mathcal{I},\nu}$,
- $\mathcal{I}, \nu \models \neg\Phi(\vec{x})$ if $\mathcal{I}, \nu \not\models \Phi(\vec{x})$,
- $\mathcal{I}, \nu \models \Phi(\vec{x}) \wedge \Psi(\vec{y})$ if $\mathcal{I}, \nu \models \Phi(\vec{x})$ and $\mathcal{I}, \nu \models \Psi(\vec{y})$,
- $\mathcal{I}, \nu \models \exists x.\Phi(\vec{y})$ if there exists $e \in \Delta^{\mathcal{I}}$ such that $\mathcal{I}, \nu[x \mapsto e] \models \Phi(\vec{y})$,

The truth value of sentences does not depend on any variable assignment; thus, assignments are omitted in this case. An interpretation $\mathcal{I}$ is a *model* of a theory $\mathcal{T}$, denoted $\mathcal{I} \models \mathcal{T}$, if $\mathcal{I}$ satisfies all sentences of $\mathcal{T}$. ◊

The distinction between a theory and a sentence in first-order logic is a more conceptual one if we focus on finite theories. In fact, semantically, a finite first-order theory $\mathcal{T}$ is equivalent to a sentence $\bigwedge_{\alpha \in \mathcal{T}} \alpha$, which we denote as $\mathsf{sen}(\mathcal{T})$.

Definition 2.6 (validity, consistency) A sentence α is *valid* if it is satisfied by all interpretations and *invalid*, otherwise. A theory $\mathcal{T}$ is *valid* if all sentences of $\mathcal{T}$ are valid. A theory $\mathcal{T}$ is *consistent (or satisfiable)* if it has a model and *inconsistent (or unsatisfiable)*, otherwise. ◊

We now define different types of reasoning services that are of interest. In a broad sense, reasoning is about entailing implicit consequences from a given set of facts, and it is therefore crucial to understand the notion of *logical consequence*.

Definition 2.7 (entailment, consequence) Let $\mathcal{T}$, $\mathcal{T}'$ be first-order theories. Then, $\mathcal{T}$ *entails* $\mathcal{T}'$, written $\mathcal{T} \models \mathcal{T}'$, if for all models $\mathcal{I}$ of $\mathcal{T}$ it holds that $\mathcal{I} \models \mathcal{T}'$. In this case, $\mathcal{T}'$ is a *consequence* of $\mathcal{T}$. ◊

With a slight abuse of notation, we write $\mathcal{T} \models \alpha$ instead of $\mathcal{T} \models \{\alpha\}$, where $\{\alpha\}$ denotes a singleton theory containing the sentence α. Classical reasoning problems are formulated as decision problems, i.e., problems that have a yes/no answer. Formulated

as a decision problem; the *entailment problem* is to decide whether $\mathcal{T} \models \mathcal{T}'$ for a given theory $\mathcal{T}$ and $\mathcal{T}'$ of first-order logic. Analogously, for a given theory $\mathcal{T}$, the *consistency problem* is to decide whether the theory $\mathcal{T}$ is consistent; the *validity problem* is to decide whether the theory is valid. Non-entailment (complement of entailment), inconsistency and invalidity checking are then the dual problems, respectively.

In essence, these reasoning problems are all tightly connected. A theory $\mathcal{T}$ is inconsistent if and only if $\mathcal{T} \models \bot$. Conversely, the entailment $\mathcal{T} \models \mathcal{T}'$ holds if and only if the theory $\{\neg \mathsf{sen}(\mathcal{T})\} \cup \mathcal{T}'$ is inconsistent. A theory $\mathcal{T}$ is valid if and only it is entailed by the empty theory, denoted $\{\} \models \mathcal{T}$, or $\models \mathcal{T}$, in short. Due to such well-known correspondences, we will mainly focus on the entailment problem in the sequel.

Concepts such as satisfiability and validity are purely semantic features, and so is the entailment relation. This, however, is not the only possible point of view. In the end, formulas are syntactic objects, and it is possible and desirable to define a calculus for deriving conclusions from the given premises that is purely syntactic, in order to allow for *automated reasoning*. These are known as proof procedures for first-order logic; among these, *semantic tableux* (Smullyan 1968), *resolution* (Robinson 1965), *natural deduction* (Gentzen 1935; Stanisław 1934) and Hilbert-style proof systems (Hilbert 1923) are well-known. In a (Hilbert-style) *deductive system (or proof system)*, deduction is a finite sequence of formulas, in which each formula is either an axiom in the theory or is obtained from previous formulas by a *rule of inference*. It is beyond the scope of this work to provide details on these calculi; for an in-depth analysis of these methods, we refer the reader to (Fitting 1996). On the other hand, no matter which proof procedure is considered, there are certain properties that are highly desired to hold. Most importantly, the proof procedure should match the semantics of the logic, which is characterized by soundness and completeness properties of the proof calculi.

Definition 2.8 (soundness, completeness) Let $\mathcal{T}$, $\mathcal{T}'$ be first-order theories and $\vdash$ denote a deduction system for first-order logic. Then, the deduction system $\vdash$ is *sound* if $\mathcal{T} \vdash \mathcal{T}'$ implies $\mathcal{T} \models \mathcal{T}'$. Conversely, the deduction $\vdash$ is *complete* if $\mathcal{T} \models \mathcal{T}'$ implies $\mathcal{T} \vdash \mathcal{T}'$ $\Diamond$

Intuitively, for a proof procedure, soundness means that all the proofs are also semantic consequences, and completeness means that all semantic consequences have proofs. Verification of soundness is rather easy as compared to completeness of a proof system for first-order logic, and the first completeness result is given in (Gödel 1929).

Theorem 2.9 (soundness and completeness of FOL) *There exists a sound and complete deduction system for first-order logic. More formally, let $\mathcal{T}$, $\mathcal{T}'$ be first-order theories. Then, there exists a deduction system $\vdash$ such that*

$$\mathcal{T} \models \mathcal{T}' \text{ if and only if } \mathcal{T} \vdash \mathcal{T}'.$$

Although first-order logic admits a sound and complete calculus, it is well known that any sound and complete calculi for first-order logic may require an infinite amount of time, in general. Thus, such a procedure may not terminate as we discuss in more depth in Section 2.3. This has motivated the study of fragments of first-order logic, of which, there are a plethora.

Fragments of First-Order Logic

In this thesis, we are concerned with several fragments of first-order logic; most of which result from i) syntactic, ii) semantic, or iii) domain-specific restrictions. We have already identified some well-known syntactic fragments of the class of first-order logic formulas in Definition 2.4. The precise correspondence between the formalisms studied in the thesis and first-order logic is to be presented in the remaining chapters in detail.

Informally, databases can be viewed as first-order interpretations over a fixed, finite domain operating under the closed-world assumption. On the other hand, ontology languages typically employ the open-world assumption and do not pose any restrictions on the domain, while being mostly syntactic fragments of first-order logic. Thus, first-order logic serves as a general unifier for the understanding of the different notions presented in the thesis.

We note, however, that fragments of first-order logic can differ significantly from each other. Especially, first-order logic restricted to finite structures (Libkin 2004), as in database theory, can make a significant difference. Many key classical theorems and techniques break down when restricted to finite structures. Conversely, a great many results, which are native to finite model theory have no classical counterpart.

We assumed familiarity with *propositional logic* and *quantified Boolean logic*. Nevertheless, each of these formalisms are essentially very restricted cases of first-order logic. Propositional logic is a special case, where the set of predicates consists of only 0-arity predicates (called propositions) and no logical quantifier is allowed. To distinguish from first-order representations, we denote propositions in lower case letters such as x, y. Similarly, we use small case Greek letters such as φ and ψ to refer to propositional formulas. Quantified Boolean logic, on the other hand, is an extension of propositional logic that also allows logical quantification over propositions with the usual semantics.

2.2 Probability Theory

Traditionally, logical reasoning takes place in a deterministic universe, where any given variable is assigned a certain value, and any statement is either *true* or *false*, with no intermediate truth value. However, for a variety of reasons, both within mathematics and logics, it is often desirable to have a rigorous framework, in which one can form uncertain statements; that is, statements, which are not true or false with definite certainty, but hold only with an intermediate degree.

Probability theory is the most widely adopted mathematical framework to formally capture the concept of uncertainty (Kolmogorov 1933). The basic intuition behind the probability theory is the notion of randomness. Imagine, for instance, an experiment, that can possibly produce a number of outcomes, while the precise outcome can not be determined with certainty. In probability theory, such randomness can be captured through *random variables* (allowing assignments to variables at random) and *events* (statements which can be true or false with some degree).

At a purely formal level, probability theory can be seen as a branch of measure theory. Following a measure-theoretic perspective, it is possible to define a *sample space*, that is, the set of outcomes of an experiment, and then model events as measurable subsets of this sample space, and random variables as measurable functions on this sample space.

Table 2.1: Worlds over the random variables $V_h = \{\mathsf{F}, \mathsf{S}, \mathsf{C}\}$. P specifies a joint probability distribution over V_h, as shown on the right-most column.

Worlds	F	S	C	$\mathrm{P}(\omega_i)$
ω_0	0	0	0	0.441
ω_1	0	0	1	0.049
ω_2	0	1	0	0.105
ω_3	0	1	1	0.105
ω_4	1	0	0	0.012
ω_5	1	0	1	0.018
ω_6	1	1	0	0.081
ω_7	1	1	1	0.189

In this way, it is possible to obtain a probability distribution over the set of outcomes (of an experiment), which, in turn, allows us to reason over uncertain situations.

In this work, we purely focus on spaces that are finite and are thus completely characterized through a finite set of *random variables*. Given these assumptions on the space, we are able to provide the basics of probability theory relevant to our purpose without the need for giving a detailed overview on measure theory; for a broader introduction to probability theory, we refer to the standard literature in the field (Durrett 2010).

We only focus on *discrete* random variables; in particular, on Bernoulli random variables. It is worth noting, however, that the restriction to Bernoulli random variables, is mainly for ease of presentation, and it can easily be relaxed to capture any discrete random variable. In the following, we will deliberately use the term random variable in place of Bernoulli random variable without further notice.

Definition 2.10 (world, sample space, event space) Let V be a finite set of random variables. A *possible world* ω over V is a *valuation* over V, which maps every random variable to a single truth value. The *sample space* Ω over V is the set of all worlds over V. The *event space* $\mathscr{A}$ over V is the powerset of the sample space Ω. $\lozenge$

Intuitively the sample space describes all the possible outcomes, or possible worlds, of all the sources of randomness and an event is simply a set of outcomes. From a broader perspective, one is typically interested only in *measurable* event spaces defined over some sample space. Nevertheless, it is easy to see that the sample and event space, as they are defined here, trivially form a measurable space since they are finite.

Example 2.11 Consider a set of random variables $V_h = \{\mathsf{F}, \mathsf{S}, \mathsf{C}\}$, where, the random variables F, S, and C are designated to model measurements of *fever*, *systolic pressure*, and *cholesterol*, respectively. More precisely, for some patient, $\mathsf{F} = 1$, denotes having high fever, $\mathsf{S} = 1$, denotes having high systolic pressure, and $\mathsf{C} = 1$ denotes having high cholesterol. The worlds over V_h are then as shown in Table 2.1. Using set notation, we can alternatively represent worlds as follows

$$\omega_0 = \{\mathsf{F} = 0, \mathsf{S} = 0, \mathsf{C} = 0\}, \ \ldots, \ \omega_7 = \{\mathsf{F} = 1, \mathsf{S} = 1, \mathsf{C} = 1\}.$$

The sample and event space is then given as

$$\Omega_{\text{ex}} = \bigcup_{0 \leq i \leq 7} \{\omega_i\} \quad , \quad \mathscr{A}_{\text{ex}} = 2^{\Omega_{\text{ex}}},$$

respectively. For instance, the set $\{\omega_6, \omega_7\} \in \mathscr{A}_{\text{ex}}$ defines an event, which is satisfied by the worlds ω_6 and ω_7. In more concrete terms, this event encodes the worlds where the events high fever ($\mathsf{F} = 1$) *and* high systolic pressure ($\mathsf{S} = 1$) are observed, or realized. $\Diamond$

A probability measure is a function, which maps each event in a measurable event space to a value in $[0, 1]$, while at the same time, satisfying certain properties.

Definition 2.12 (probability measure) Let $\mathscr{A}$ be a non-empty set. A *probability measure* is a function $\mathrm{P} : \mathscr{A} \mapsto [0, 1]$ such that:

- $\mathrm{P}(\emptyset) = 0$,
- $\mathrm{P}(\mathscr{A}) = 1$,
- $\mathrm{P}(\bigcup_i \varepsilon_i) = \sum_i \mathrm{P}(\varepsilon_i)$ for countable disjoint $\varepsilon_i \in \mathscr{A}$. (countably additive) $\Diamond$

We introduced the notions of sample and event space, which are completely characterized by random variables, and defined the concept of probability measure over such spaces. Putting all together, we are speaking of probability spaces.

Definition 2.13 (probability space) Let Ω be the sample space, and $\mathscr{A}$ the event space defined over a finite set of random variables V. Then, the triple $(\Omega, \mathscr{A}, \mathrm{P})$, where P is a probability measure over $\mathscr{A}$, forms a *probability space*. $\Diamond$

We specify the probability space of interest with the help of a finite set of random variables and view events as sets of worlds. As there can be exponentially many worlds in the number of the random variables, in order to write an event of interest, one has to enumerate all worlds that define this event. This is certainly not the best possible representation for events. An alternative is to form an event language to be able to compactly represent sets of worlds. We, therefore, view events as propositional formulas, built inductively from *elementary events*, using logical connectives.

Definition 2.14 (event language) Let V be a set of random variables, an *elementary event* over V is either of the form $X = 1$, or of the form $X = 0$, where $X \in V$. An *event* θ is then inductively defined by the grammar rule

$$\theta \quad := \quad \mathsf{e} \quad | \quad \neg\theta \quad | \quad \theta \wedge \theta \quad ,$$

where e is an elementary event over V. $\Diamond$

As usual, disjunction ($\vee$) and implication ($\rightarrow$) are used as syntactic sugar in the language. Sometimes, we write ", " in place of " $\wedge$ ", if there is no danger of ambiguity.

In essence, events are propositional formulas defined over a set of random variables. As a consequence of this, we can identify a world that an event encodes through the (propositional) entailment relation: a world ω satisfies an event θ, if and only if $\omega \models \theta$.

The probability spaces defined with the help of a finite set of discrete random variables, are essentially discrete. Therefore, the probability measure P induces a *joint discrete*

probability distribution over the valuations of the set of random variables. There are numerous ways of representing joint probability distributions; a naïve approach is to enumerate all the worlds along with a probability value as shown in Table 2.1. The probability of an arbitrary event θ can then be computed from the given joint probability distribution as

$$\mathrm{P}(\theta) \coloneqq \sum_{\omega \models \theta} \mathrm{P}(\omega),$$

by summing over the probabilities of the worlds ω that satisfy the formula θ. Specifically, events can be seen as *queries* posed on top of a probabilistic model, which can hold with some probability. Often, while posing the query, one may also have some new *evidence* about the world; such evidence, itself, can also be formulated as an event. To incorporate such observations, or evidence, one studies the *conditional probability* of a (query)-event given an (evidence)-event.

Definition 2.15 (conditional probability) The *conditional probability* of an event θ_1 given another event θ_2 is given by

$$\mathrm{P}(\theta_1 \mid \theta_2) \coloneqq \frac{\mathrm{P}(\theta_1 \wedge \theta_2)}{\mathrm{P}(\theta_2)},$$

where $\mathrm{P}(\theta_2) > 0$. ◊

Conditional dependencies constitute natural, real-world phenomena, which can be observed in different domains. For instance, the probability of observing a *thunderstorm* on a sunny day is relatively low, compared to a rainy day. As a completely different example, people with high *hypercholesterolemia* are more likely to have a heart attack compared to people without this condition.

With the help of conditional probabilities, we also obtain another way of specifying a joint probability distribution. Instead of assigning a probability to each world, we specify the conditional probabilities over elementary events, which is then sufficient to define a joint probability distribution. As we will elaborate in depth later (in the context of probabilistic graphical models), such representations are much more succinct once combined with some additional assumptions.

Definition 2.16 (joint probability distribution) Let $V = \{X_1 \ldots, X_n\}$ be a finite set of random variables and Θ a set of conditional probabilities $\mathrm{P}(X_i \mid X_{i+1}, \ldots, X_n)$ for all valuations of the random variables X_i, $1 \leq i \leq n$. The *joint probability distribution* over V is then given by the chain rule

$$\mathrm{P}(X_1, \ldots, X_n) \coloneqq \prod_{i=1}^{n} \mathrm{P}(X_i \mid X_{i+1}, \ldots, X_n),$$

from the given conditional probabilities. ◊

It is thus possible to recover a probability distribution from a given set of conditional probabilities. Besides, encoding conditional dependencies explicitly helps us to filter the important parameters for our query.

Example 2.17 Consider our running example. P specifies a probability distribution over the valuations of random variables in V_h

$$\begin{aligned} \mathrm{P}(\omega_0) &= \mathrm{P}(\mathsf{F}=0,\mathsf{S}=0,\mathsf{C}=0) = 0.041, \\ &\dots \\ \mathrm{P}(\omega_7) &= \mathrm{P}(\mathsf{F}=1,\mathsf{S}=1,\mathsf{C}=1) = 0.189, \end{aligned}$$

as given in Table 2.1. For example, the conditional probability $\mathrm{P}(\mathsf{F}=0 \mid \mathsf{S}=0)$, i.e., the probability of the event $\mathsf{F}=0$ given $\mathsf{S}=0$, can be computed as

$$\frac{\mathrm{P}(\mathsf{F}=0 \wedge \mathsf{S}=0)}{\mathrm{P}(\mathsf{S}=0)},$$

which is given by

$$\frac{\mathrm{P}(\omega_0)+\mathrm{P}(\omega_1)}{\mathrm{P}(\omega_0)+\mathrm{P}(\omega_1)+\mathrm{P}(\omega_4)+\mathrm{P}(\omega_5)} = \frac{0.49}{0.52} \approx 0.94.$$

In practical terms, this means that whenever the patient has high systolic pressure, it is highly likely (94%) that the patient also has high fever. In this completely synthetic example, having high fever is strongly associated with having high systolic pressure. ◊

In general, the conditional probability of an event θ_1 given an event θ_2 is not equal to $\mathrm{P}(\theta_1)$, i.e., the unconditional probability of θ_1, as already illustrated in the example. The special case where $\mathrm{P}(\theta_1 \mid \theta_2) = \mathrm{P}(\theta_1)$, we say that θ_1 is *independent* of θ_2; more precisely, θ_1 is independent of θ_2 if observing θ_2 does not change the probability of θ_1.

Definition 2.18 (independence) Let $\{\theta_1, \dots, \theta_n\}$ be a finite set of events. These events are *(mutually) independent* if it holds that

$$\mathrm{P}(\theta_1 \wedge \dots \wedge \theta_n) = \mathrm{P}(\theta_1) \times \dots \times \mathrm{P}(\theta_n).$$

A set $V = \{X_1 \dots, X_n\}$ of random variables are said to be *(mutually) independent* if

$$\mathrm{P}(X_1, \dots, X_n) = \mathrm{P}(X_1) \times \dots \times \mathrm{P}(X_n),$$

holds for all valuations of the random variables X_i. ◊

In the remainder of this text, we will specify the probability space through a set of random variables, as explained here.

2.3 Complexity Theory

This section briefly introduces basic elements of computational complexity theory. We first recall basics of the theory of computation based on Turing machines. This is followed by a discussion on Turing complexity classes and different reducibility notions. Afterwards, the theory of computation is being revisited with an alternative computation model that is based on circuits, followed by an overview of circuit complexity classes and the respective reducibility notions. For an in-depth coverage of the theory of computation as well as complexity theory, we refer the reader to the excellent books in the literature (Arora and Barak 2009; Hopcroft and Ullman 1979; Papadimitriou 1994; Sipser 1996).

2.3.1 Turing Machines

The mathematical modeling of computation is achieved through a simple, and yet very powerful model, i.e., the Turing machine, as first proposed by Alan Turing (Turing 1936). A Turing machine is very similar to a *finite automaton*, with the difference of having an unlimited memory. Formally, a (deterministic) Turing machine is defined as follows.

Definition 2.19 A *(deterministic) Turing machine (TM)* is a tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ where Q, Σ, Γ are all finite sets and

- Q is the set of *states*,
- Σ is the *input alphabet*, which does not contain the *blank symbol* $\sqcup$,
- Γ is the *tape alphabet*, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$ is a partial function, called the *transition function*, where L is left shift, R is right shift.
- $q_0 \in Q$ is the *starting state*,
- $q_a \in Q$ is an *accepting state*,
- $q_r \in Q$ is a *rejecting state*. ◊

A Turing machine can *read* from an input tape and *write* to an output tape. Given an input string, it operates on finite number of states in accordance to a transition function that specifies the actions of moving the head right (R), or left (L). The computation continues until the TM enters an *accepting* or *rejecting* state; such states are also called a *Halting state*. A *nondeterministic* TM is defined in the same way as a deterministic TM except the fact that the transition function is not deterministic anymore:

$$\delta : Q \times \Gamma \mapsto \mathbb{P}(Q \times \Gamma \times \{L, R\}).$$

Intuitively, a deterministic TM specifies a single computation path, while a nondeterministic TM can specify exponentially many such paths. As a consequence, a nondeterministic TM serves as a more succinct model. Clearly, this does not imply that it is more powerful than a deterministic TM. In fact, it is well-known that any nondeterministic TM can be simulated with a deterministic one; obviously, at potential cost of an exponentially larger representation. For other types of TMs, and the relations among them, we refer the reader to (Sipser 1996).

The collection of strings accepted by a TM M is the *language* of M, denoted $\mathcal{L}(M)$. If $\mathcal{L}(M)$ for a TM M, then $\mathcal{L}$ is said to be *Turing-recognizable (or recursively enumerable)*. Notice that there are three possibilities for computations of a TM M on an arbitrary input: M either *accepts*, or *rejects*, or enters an *infinite loop* on an input (i.e., it never reaches a Halting state). A restricted model of TMs is the *decider (or halting)* TM, which halt on all inputs by accepting or rejecting. A language is called *decidable (or recursive)* if some TM *decides* it and *undecidable*, otherwise.

As defined here, a decider Turing machine can only accept or reject; that is, its output is always binary. This definition can be generalized to Turing machines that can also output a string as in (Sipser 1996).

Definition 2.20 A function $f : \Sigma^* \mapsto \Sigma^*$ is *computable* if some Turing machine M, on every input w, halts with just $f(w)$ on its tape. ◊

The connection between decision problems and languages is obvious: a decision problem can be specified in terms of a language. Thus, we will generally speak problems instead of languages.

A canonical example of an undecidable problem is the *halting problem*: given a TM M and an input w, does M halt on w? The undecidability of the halting problem is typically shown using Cantor's (simple and yet very powerful) diagonalisation argument (Cantor 1891). Although being Turing-recognizable, the halting problem is an undecidable problem, implying that Turing machines are more powerful than decider machines. Nevertheless, some languages are not even Turing-recognizable, which is a consequence of the fact that there are uncountably many languages and only countably many TMs.

Recall that, for the entailment problem, there is a sound and complete deduction system in first-order logic. However, we deferred to this section whether such a procedure could at the same time be terminating. By independent results of Church and Turing, any sound and complete deduction system for fist order logic is non-terminating.

Theorem 2.21 (Church and Turing 1936) *Entailment in first-order logic is undecidable.*

As pointed out before, many results in first-order logic do not hold when restricted to finite structures. Unfortunately, this is not the case for entailment, which remains undecidable even over finite structures.

Theorem 2.22 (Trakhtenbrot 1950) *Entailment in first-order logic over finite structures is undecidable.*

In contrast to these undecidability results, many fragments of first-order logic have already been identified to be decidable; see (Börger, Grädel, and Gurevich 1997), for an excellent review. Nevertheless, decidability does not necessarily imply that the computational problem can be solved in practice, as the amount of resources, such as *time* and *space*, is very much limited in the real-world. This is the driving force behind the study of computational complexity theory.

Time and Space Complexity. Computational complexity theory is about classifying computational problems according to the resources needed to solve the problem, providing much more refined views about the computational difficulty of the problem. We, therefore, first recall how time and space is measured based on the underlying computational model of Turing machines.

Definition 2.23 Let M be a deterministic TM that halts on all inputs. The *time complexity* of M is the function $f : \mathbb{N} \mapsto \mathbb{N}$, where $f(n)$ is the maximum number of steps that M uses on any input of length n. ◊

Using standard conventions, we call a TM M, a $f(n)$-time bounded TM if M runs in time $f(n)$. More generally, we speak of polynomial-time TMs if $f(n)$ is a polynomial function and we use the same terminology for other classes, such as exponential functions. Space complexity is then defined analogously.

Definition 2.24 Let M be a deterministic TM that halts on all inputs. The *space complexity* of M is the function $f : \mathbb{N} \mapsto \mathbb{N}$, where $f(n)$ is the maximum number of tape cells (memory) that M uses on any input length n. ◊

We use the same naming conventions as in the time complexity and speak of polynomial space TMs, exponential space TMs, et cetera. Finally, time and space complexity for nondeterministic Turing machines can be defined analogously.

We now recall some complexity classes, which are defined based on these measures. Our analysis is separated in two parts; first, we present decision-theoretic complexity classes (and oracle machines) and then functional complexity classes.

Decision-Theoretic Complexity Classes. Decision problems have either *yes* or *no* as answers; by decision-theoretic complexity classes, we refer to classes that encompass only decision problems. Perhaps the most well-known complexity class is polynomial time (P), which captures precisely those problems that can be decided on a polynomial time Turing machine. P is widely considered as the class of problems that are realistically solvable on a computer and such problems are sometimes also called *tractable*.

Definition 2.25 P is the class of languages that can be decided by a polynomial time deterministic Turing machine. ◊

The nondeterministic variant of P, the class NP, encompasses many other interesting problems, which are not known to be in P.

Definition 2.26 NP is the class of languages that can be decided on a polynomial time nondeterministic Turing machine. ◊

Knowing that a problem is in NP, alone, is not very satisfactory as it does not imply anything regarding the *difficulty* of the problem as the lower bound of the complexity of the problem remains open. The notion of *reduction* plays a key role in this respect, as it provides us with the right means to determine the *difficulty* of a problem, i.e., by relating it to the difficulty of other problems. For showing NP-hardness, *polynomial time many-one reductions* are *de facto standard* reductions. Many-one reductions date back to (Post 1944) and (Shapiro 1956); polynomial time many-one reductions are due to Karp and are thus also know as Karp reductions (Karp 1973).

Definition 2.27 A language $\mathcal{L}_1$ is *polynomial time many-one reducible* to a language $\mathcal{L}_2$, written $\mathcal{L}_1 \leq_{\mathrm{P}} \mathcal{L}_2$, if there exists a polynomial time computable function $f : \Sigma^* \mapsto \Sigma^*$, where for every input w, it holds that

$$w \in \mathcal{L}_1 \quad \textit{if and only if} \quad f(w) \in \mathcal{L}_2.$$

The function f is called polynomial time reduction of $\mathcal{L}_1$ to $\mathcal{L}_2$. ◊

We say that a language $\mathcal{L}$ is NP-*complete* if $\mathcal{L} \in$ NP and every language $\mathcal{H} \in$ NP is polynomial time many-one reducible to $\mathcal{L}$. Throughout the thesis, we always assume polynomial time many-one reductions unless explicitly stated otherwise. Besides, the concept of completeness generalizes to other decision-theoretic complexity classes in the obvious way.

(Cook 1971) and (Levin 1973) proved the first NP-complete problem, namely, satisfiability of propositional formulas (SAT) by a direct reduction from a nondeterministic Turing machine, which paved the way for many other NP-complete problems. For

example, 3SAT, that is, the problem of deciding satisfiability of propositional formulas in 3CNF, is NP-complete.

The complexity class PP (Gill 1977) is very central for problems related to counting and is thus relevant for the complexity of probabilistic inference tasks. For example, probabilistic inference in Bayesian networks is PP-complete (Littman, Majercik, and Pitassi 2001), as discussed in Chapter 5.

Definition 2.28 PP is the set of languages recognized by a polynomial time nondeterministic Turing machine that accepts an input if and only if *more than half* of the computation paths are accepting. ◊

There are alternative definitions of PP based on probabilistic Turing machines, which are equivalent to the given definition. The canonical problem for PP is MAJSAT; that is, given a propositional formula φ, the problem of deciding whether the majority of the assignments to φ are satisfying. Note that majority serves as a default threshold value, but this threshold can be modified by introducing artificial success and failure branches into the nondeterministic Turing machine. In other words, it is possible to decide whether the number of satisfying assignments of φ are greater than some threshold value in PP. Obviously, this is a generalization of NP, which searches for a single satisfying assignment, i.e., a single solution. More PP-complete problems related to propositional satisfiability can be found in (Bailey, Dalmau, and Kolaitis 2007).

Importantly, PP is closed under *truth table reductions* (Beigel, Reingold, and Spielman 1995); in particular, this implies that PP is closed under *complement*, *union* and *intersection*. PP contains NP and is contained in PSPACE which is defined next.

Definition 2.29 PSPACE is the class of languages that can be decided by a polynomial space deterministic Turing machine. ◊

For example, checking whether an arbitrary quantified Boolean formula is valid is a canonical PSPACE-complete problem. NPSPACE is the nondeterministic variant of PSPACE that defines the class of languages that can be decided by a polynomial space nondeterministic Turing machine. Savitch has proven that PSPACE =NPSPACE (Savitch 1970). The classes EXP and NEXP are then the exponential time analogs of P and NP, respectively, which are defined as follows.

Definition 2.30 EXP (resp., NEXP) is the class of languages that can be decided by an exponential time deterministic Turing machine (resp., nondeterministic Turing machine). ◊

An example of an NEXP-complete problem is the satisfiability of first-order formulas from the two variable fragment of first-order logic as shown in (Grädel, Kolaitis, and Vardi 1997). Checking satisfiability of Schönfinkel–Bernays formulas, that is, formulas of the form $\exists\forall.\Phi$, is also a well-known example of a NEXP-complete problem (Lewis 1980). The classes 2EXP, N2EXP, … are defined analogously. The relations between these complexity classes can be summarized as follows

$$\text{P} \subseteq \text{NP} \subseteq \text{PP} \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXP} \subseteq \text{NEXP},$$

where only the containments $\text{P} \subset \text{EXP}$ and $\text{NP} \subset \text{NEXP}$ are known to be strict, both of which follow from the *time hierarchy theorem* (Cook 1972).

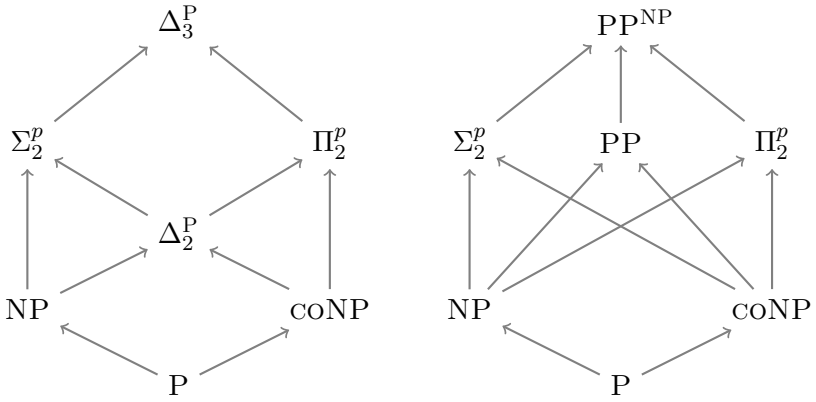


Figure 2.1: A portion of the polynomial time hierarchy is shown on the left-hand side where arrows denote inclusion relationship. On the right-hand side, the relationship of PP and $\mathrm{PP}^{\mathrm{NP}}$ to the first level of the PH is depicted.

Oracle Machines and Hierarchies. Oracle machines constitute another interesting phenomenon in computational complexity. Intuitively, an oracle is a black box, which is able to produce an answer to a problem. Many interesting problems can be captured by Turing machines that have an oracle access.

Definition 2.31 An *oracle* for a language $\mathcal{L}$ is an external device that is capable of reporting whether for any string $w \in \mathcal{L}$. An *oracle Turing machine* is a Turing machine with an *oracle tape* and additional states q_{query}, q_{yes} and q_{no}. Let w denote the string in the oracle tape of the Turing machine. Then, whenever the Turing machine enters the state q_{query}, it moves to state q_{yes} if the oracle accepts w and it moves to state q_{no} otherwise. We write $M^{\mathcal{L}}$ to denote a Turing machine M with an access to an oracle for the language $\mathcal{L}$. ◊

If the language $\mathcal{L}$ in the oracle of $M^{\mathcal{L}}$ is *complete* for a class $\mathfrak{C}$, we write $M^{\mathfrak{C}}$. We next define the polynomial hierarchy, which contains many natural problems (Stockmeyer 1976).

Definition 2.32 Let $\Delta_0^{\mathrm{P}} = \Sigma_0^{\mathrm{P}} = \Pi_0^{\mathrm{P}} = \mathrm{P}$. Then, the *polynomial hierarchy* (PH) is given as

$$\Delta_{i+1}^{\mathrm{P}} = \mathrm{P}^{\Sigma_i^{\mathrm{P}}}, \quad \Sigma_{i+1}^{\mathrm{P}} = \mathrm{NP}^{\Sigma_i^{\mathrm{P}}}, \quad \Pi_{i+1}^{\mathrm{P}} = \mathrm{coNP}^{\Sigma_i^{\mathrm{P}}}$$

for some $i \geq 0$. ◊

For example, given a quantified Boolean formula of the form $\exists \vec{x}\, \forall \vec{y}.\varphi$, deciding whether it is valid is Σ_2^{P}-complete. Conversely, if we consider formulas of the form $\forall \vec{x}\, \exists \vec{y}.\varphi$, then the decision problem is Π_2^{P}-complete.

Consider now a nondeterministic Turing machine with an NP oracle, which answers *yes* if and only if majority of its runs are accepting. This precisely describes the complexity class $\mathrm{PP}^{\mathrm{NP}}$. Intuitively, for every run, to decide whether it is an accepting run, such a machine can use the power of the NP oracle. The relation of PP and $\mathrm{PP}^{\mathrm{NP}}$ to the PH (along with a portion of the PH) is depicted in Figure 2.1.

Another class of interest is $\mathrm{NP}^{\mathrm{PP}}$, which intuitively combines search and optimization problems (in the reverse direction of $\mathrm{PP}^{\mathrm{NP}}$). A natural canonical problem for this class is EMAJSAT (Littman, Goldsmith, and Mundhenk 1998); that is, given a propositional formula φ and a set of distinguished variables $\vec{x}$ from φ, is there an assignment μ to $\vec{x}$-variables such that majority of the assignments τ that extend μ satifies φ. $\mathrm{NP}^{\mathrm{PP}}$ appears as a very fundamental class for probabilistic inference and planning tasks. For instance, *maximal posterior inference* in Bayesian Networks is $\mathrm{NP}^{\mathrm{PP}}$-complete (Park and Darwiche 2004b).

Similar to the polynomial hierarchy, it is possible to define a counting hierarchy (CH), which includes classes such as $\mathrm{PP}^{\mathrm{PP}}$; for a logical characterization and for canonical problems in this hierarchy, please see (Torán 1991; Wagner 1986).

Functional Complexity Classes. All the complexity classes considered so far are of a decision-theoretic nature. On the other hand, many problems require answers which are not necessarily binary. Decision problems ask whether a solution exists, whereas functional, counting problems ask how many different solutions exist. Functional complexity classes capture this type of problems. We introduce FP, which is the functional analog of the decision-theoretic complexity class P.

Definition 2.33 FP is the class of functions $f : \{0,1\}^* \mapsto \{0,1\}^*$ computable by a polynomial time deterministic Turing Machine. ◊

Valiant introduced the class #P (Valiant 1979a), and proved that the computation of the *permanent* of 0-1 matrices is #P-complete, which is in sharp contrast with the computation of the determinant. Note that the definition of an oracle TM (see Definition 2.31) is generalized to a functional oracle such that it is also able to query and retrieve non-binary values.

Definition 2.34 A function $f : \{0,1\}^* \mapsto \mathbb{N}$ is in #P if there exists a polynomial $p : \mathbb{N} \mapsto \mathbb{N}$ and a polynomial time deterministic Turing machine M such that for every $x \in \{0,1\}^*$, it holds that

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid M \textit{ answers } y \textit{ on the input } x\}| \qquad \Diamond$$

The canonical problem for #P is #SAT; that is, given a propositional formula φ, the task of computing the number of satisfying assignments to φ.

Several types of reductions exist for #P (and for classes beyond #P), while the most common being the so-called *polynomial time Turing reductions*, also known as Cook reductions (Cook 1971). Informally, Turing reductions generalize many-one reductions in the sense that they also allow access to an oracle.

Definition 2.35 A function f is #P-complete under polynomial time Turing reductions if it is in #P and every $g \in \#\mathrm{P}$ is in FP^f. ◊

Turing reductions are defined similarly for decision-theoretic complexity classes. Nevertheless, Turing reductions are rather uncommon for these classes. For instance, the PH collapses to the first level under polynomial time Turing reductions, which is widely believed not to be true. Therefore, many-one reductions give rise to a richer, more delicate, hierarchy for decision-theoretic classes.

Turing reductions have also been criticized in the context of functional complexity classes, as many classes above #P appear to be not closed under polynomial time Turing reductions (Durand, Hermann, and Kolaitis 2005). Alternatively, *parsimonious reductions* (Simon 1977) are known, which require the number of solutions of the problem to be preserved in the reduction. Unfortunately, this is a very strong condition and can not be satisfied for many problems. *Subtractive reductions*, as proposed in (Durand, Hermann, and Kolaitis 2005), relax this condition as they make it possible to count the number of solutions by first overcounting them and then carefully subtracting any surplus. Several interesting problems, such as counting the number of perfect matchings in a bipartite graph, are known to be #P-hard under polynomial time Turing reductions, while it is still an open problem whether this also holds under subtractive reductions. For detailed discussions on comparisons of different types of reductions, see (Durand, Hermann, and Kolaitis 2005; Ladner, Lynch, and A. Selman 1975).

In this work, #P-hardness is with respect to polynomial time Turing reductions, unless explicitly specified otherwise. (Valiant 1979b) has shown several #P-complete problems under Turing reductions. Notably, in Valiant's words, there are problems, which are *easy to decide hard to count*: for example, satisfiability of a propositional formula in 2CNF can be decided in polynomial time, whereas counting the number of models of a propositional monotone formula in 2CNF is already #P-hard under these reductions (Provan and Ball 1983).

On The Relations Between PH, PP and #P. Intuitively, the class PP can be seen as the decision counterpart of the counting class #P. (Toda and Watanabe 1992) prove an inclusion result, which asserts that $\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}}$; that is, a polynomial time deterministic Turing machine with access to a #P oracle is capable of deciding all problems in the polynomial hierarchy. Their result holds even under 1-Turing reductions, i.e., even when the #P oracle can be queried only once. The close connection between PP and #P is then given by Toda's theorem, which proves $\mathrm{P}^{\mathrm{PP}} = \mathrm{P}^{\#\mathrm{P}}$ (Toda 1989). Toda's theorem is actually stronger than this well-known result: it asserts that $\mathrm{PP}^{\mathrm{PH}} \subseteq \mathrm{P}^{\mathrm{PP}}$. The relationship among these classes is given as

$$\mathrm{PP}, \mathrm{PH} \subseteq \mathrm{PP}^{\mathrm{PH}} \subseteq \mathrm{P}^{\mathrm{PP}} = \mathrm{P}^{\#\mathrm{P}} \subseteq \mathrm{NP}^{\mathrm{PP}} \subseteq \mathrm{PP}^{\mathrm{PP}} \subseteq \mathrm{CH} \subseteq \mathrm{PSpace}.$$

2.3.2 Boolean Circuits

We briefly revisit an alternative computation model for the theory of computing that is based on circuits. For a detailed treatment of the subject, please consult the relevant literature (Vollmer 1999).

For every $n \in \mathbb{N}$, a n-input, single output *Boolean circuit* is a directed acyclic graph, in which every node is either an AND, OR, NOT or a MAJORITY gate and one of these gates is designated as the *output gate*. Each gate is a processor that computes the corresponding Boolean function. The semantics of AND, OR, NOT gates are as usual. MAJORITY gates output 1 if and only if the majority of its inputs are 1. The maximal number of incoming edges for a gate is called its *fan-in*.

Note that a circuit can only capture an input of a certain, fixed size. To be able to represent languages (of arbitrary size) in terms of circuits, we need the notion of a circuit family.

Definition 2.36 A *circuit family* $\mathfrak{C}$ is an infinite list of circuits $\{C_n \mid n \in \mathbb{N}\}$, where C_n has n input variables. We say that $\mathfrak{C}$ decides a language $\mathcal{L}$ over $\{0,1\}$ if for every string w, $w \in \mathcal{L}$ if and only if $C_n(w) = 1$. ◊

The *size* of a circuit is measured in the number of the gates that it contains. A circuit is said to be *minimal in size* if there is no smaller circuit equivalent to it. A circuit family $\{C_n \mid n \in \mathbb{N}\}$ is minimal if every C_n is minimal. The *size complexity of a circuit family* $\{C_n \mid n \in \mathbb{N}\}$ is given by a function $f : \mathbb{N} \mapsto \mathbb{N}$, where the size of C_n is at most $f(n)$.

Analogously, we can define the *depth complexity of a circuit family*. The *depth* of a circuit is the depth of the underlying graph; a circuit is said to be *minimal in depth* if there is no circuit equivalent to it with smaller depth. Finally, the *depth complexity of a circuit family* $\{C_n \mid n \in \mathbb{N}\}$ is given by a function $f : \mathbb{N} \mapsto \mathbb{N}$, where the depth of C_n is at most $f(n)$.

Then, the *circuit complexity* of a language is the size complexity of a minimal circuit family for that language. The *circuit depth complexity* of a language is the depth complexity of a minimal circuit family for that language.

The types of gates and their fan-in depends on the particular circuit class. Important circuit classes are given in terms of NC, AC and TC hierarchies.

Definition 2.37 NC, AC and TC represent a different complexity class hierarchy where

- NC^i consists of the languages recognized by Boolean circuits with depth $O(\log^i(n))$ and polynomial number of *constant fan-in* AND and OR gates.
- AC^i consists of the languages recognized by Boolean circuits with depth $O(\log^i(n))$ and polynomial number of *unbounded fan-in* AND and OR gates.
- TC^i consists of the languages recognized by Boolean circuits with depth $O(\log^i(n))$ and a *polynomial number* of *unbounded fan-in* AND, OR and MAJORITY gates.

Then the hierarchies NC, AC and TC are respectively defined as

$$\mathrm{NC} \coloneqq \bigcup_i \mathrm{NC}^i \quad , \quad \mathrm{AC} \coloneqq \bigcup_i \mathrm{AC}^i \quad , \quad \mathrm{TC} \coloneqq \bigcup_i \mathrm{TC}^i.$$ ◊

Unlike Turing machines, Boolean circuits are non-uniform models of computation; that is, inputs of different size are processed by different circuits. Every member of a circuit family $\{C_n \mid n \in \mathbb{N}\}$ can be associated with a different type of input. It is therefore common to impose some *uniformity* conditions that require the existence of some resource-bounded Turing machine that, on input n, produces a description of the individual circuit C_n.

For instance, if there exists a *polynomial time* deterministic Turing machine that can produce such a circuit, then this entails polynomial time-uniformity. Unfortunately, polynomial time-uniformity is too strong when constant depth circuit classes such as AC^0 and TC^0 are considered. The most widely accepted uniformity condition for these classes is DLogTime-uniformity, bounding the computation in accordance to a logarithmic

time deterministic Turing machine. Our primary concern is regarding constant depth circuits and they are related to some well-known classes as

$$\mathrm{NC}^0 \subseteq \mathrm{AC}^0 \subset \mathrm{TC}^0 \subseteq \mathrm{NC}^1 \subseteq \textsc{LogSpace} \subseteq \textsc{NLogSpace} \subseteq \mathrm{P}.$$

An arithmetic problem that is in AC^0 is *binary addition*. Note, however, that calculating the *parity* of an input is possible with NC^1 circuits, while this problem cannot be decided by any uniform AC^0 circuit as shown in a breakthrough result by (Furst, Saxe, and Sipser 1984). Thus, the inclusion between these classes is strict. Moreover, as *majority* can be uniformly reduced to *parity*, this, in particular, implies that the majority cannot be decided by any uniform AC^0 circuit. As a consequence, the inclusion between AC^0 and TC^0 is also strict. An arithmetic problem that can not be decided in AC^0, but turns out to be complete for TC^0 under DLogTime-uniformity is binary multiplication (Hesse, Allender, and Barrington 2002).

These classes are very closely linked to descriptive complexity theory (Immerman 1999), which characterizes complexity classes by the type of logic needed to express the languages in them. Specifically, each logical system produces a set of queries expressible in it; once restricted to finite structures, these queries correspond to the computational problems of traditional complexity theory. For example, FO formulas over finite structures can be encoded into AC^0 circuits. More precisely, FO extended with the BIT operator is equivalent to AC^0. TC^0 is closely linked to FO extended with counting quantifiers as in (Wagner 1986). Besides, FO with a transitive closure operator equals NLogSpace; if furthermore, the transitive closure operator is also symmetric, it equals to LogSpace. Similar correspondences are very fundamental for database theory.

Part II

Probabilistic Databases

Chapter 3

Preliminaries, First Results and State of the Art

This chapter introduces some of the main notions used in this thesis. First a model-theoretic view on databases is presented, which is going to be relevant throughout the thesis. Subsequently, probabilistic databases are introduced; in particular, the tuple-independent probabilistic database model. An overview of existing results regarding the complexity of probabilistic query evaluation in probabilistic databases is given, which is then followed by some new results. At the end, this chapter contains a survey of related work on probabilistic databases.

3.1 Database Theory

Relational databases are *de facto standard* tools for data management: they provide sophisticated means for *storing*, *processing*, and *querying* data sources. In principle, a relational database is nothing but a *structured* collection of *things*. Having put in this generality, it is unclear how such a structured collection can be effective in storing, processing, or querying information. Thanks to the pioneering work of (Codd 1972), a mathematical model for relational databases has been formulated, known as *relational algebra*, allowing users to fetch and process specific information from a database, in a principled and sound manner.

3.1.1 The Model-Theoretic View on Databases

Intellectual roots of databases are in first-order logic (Abiteboul, Hull, and Vianu 1995); in particular, in finite model theory (Libkin 2004). Thus, we adopt the model-theoretic perspective and view databases as first-order structures over some fixed domain.

Definition 3.1 (database) Let σ be a *finite* relational vocabulary, which consists of *finite* sets $\mathbf{R}$ of *predicate*, $\mathbf{C}$ of *constant*, and $\mathbf{V}$ of *variable names*. A σ-atom is a ground atom over the vocabulary σ. A *database* $\mathcal{D}$ over a relational vocabulary σ is a finite set of σ-atoms. $\Diamond$

We usually omit the vocabulary σ from the notation, if it is clear from context. A classical representation of a relational database is in terms of database tables, which organizes sets of atoms, relative to the predicate names. Each table corresponds to a *predicate* and its rows correspond to *ground atoms* of that predicate, which are also called *records*, *facts*, or *tuples*.

Example 3.2 Let us consider a database $\mathcal{D}_m$ given in Table 3.1, in terms of three database tables, corresponding to the predicates StarredIn, DirectedBy and Awarded,

respectively. For example, the last row in the DirectedBy table is interpreted as the ground atom DirectedBy(winterSleep, ceylan). This database is represented as

$$\mathcal{D}_m \coloneqq \{\mathsf{StarredIn}(\mathsf{deNiro}, \mathsf{taxiDriver}), \ldots, \mathsf{Awarded}(\mathsf{winterSleep}, \mathsf{palmed'Or})\},$$

in set notation. We will use both notations interchangeably. ◊

The query semantics is then given as a first-order semantics, which takes the additional assumptions employed in databases also into account.

Definition 3.3 (semantics) Let σ be a *finite* relational vocabulary, which consists of *finite* sets $\mathbf{R}$ of *predicate*, $\mathbf{C}$ of *constant*, and $\mathbf{V}$ of *variable names*. A database $\mathcal{D}$ over σ defines a first-order interpretation over σ, where

i) the domain is given as $\Delta^{\mathcal{D}} = \mathbf{C}$, (closed-domain assumption)

ii) $a^{\mathcal{D}} = a$, for all constants $a \in \mathbf{C}$, (standard name assumption)

iii) $(a_1, \ldots, a_n) \in \mathsf{P}^{\mathcal{D}}$ iff $\mathsf{P}(a_1, \ldots, a_n) \in \mathcal{D}$, for all $\mathsf{P}^{\mathsf{n}} \in \mathbf{R}$, (closed-world assumption)

The *satisfaction relation* $\models$ is then defined, as before. We say that a database $\mathcal{D}$ *satisfies* a formula Φ if $\mathcal{D} \models \Phi$. ◊

Let us have a closer look at this semantics and highlight some of the important differences with classical first-order semantics, which will be critical throughout this work. The *closed-domain assumption* (CDA) restricts the domain of an interpretation to a *finite*, *fixed* set of constants; namely, to database constants. As a matter of fact, such interpretations presume that the *domain is complete*. The *standard name assumption* (SNA) ensures a bijection between the database constants and the domain: it is not possible to refer to the same individual in the domain with two different constant names. Last, but certainly not least, the *closed-world assumption* (CWA) of databases forces anything that is *not known to be true*, to be *false* (Reiter 1978); that is, databases make the *data completeness* assumption. Besides, under the closed domain and standard name assumptions, it is easy to see that the given semantics coincides with the *Herbrand semantics* (Hinrichs and Genesereth 2006). Briefly, a *Herbrand interpretation* over σ maps every σ-atom to either *true*, or *false*. Then, a database is simply a Herbrand interpretation, where the atoms that appear in the database are mapped to *true*, while ones not in the database are mapped to *false*, according to the closed-world assumption.

The simplifying assumptions of databases are useful for a variety of reasons. At the same time, it becomes very easy to produce some undesirable consequences under these assumptions as we will elaborate. We will revisit some of these assumptions, and discuss their implications, in depth, in the sequel, and illustrate our major motivation from this perspective.

The most fundamental task in databases is *query answering*; that is, given a database $\mathcal{D}$ and a formula $\Phi(x_1, \ldots, x_n)$ of first-order logic, to decide whether there exists assignments to free variables $x_1, \ldots, x_n$, such that the resulting formula is satisfied by the database.

Importantly, here the variable assignments are of a special type, also called *substitutions*. Formally, a *substitution* $[x/t]$ replaces all occurrences of the variable x by some database constant t in some formula $\Phi[x, y]$, denoted $\Phi[x/t]$.

Table 3.1: A database $\mathcal{D}_m$ represented in terms of relational database tables.

StarredIn	
deNiro	taxiDriver
foster	taxiDriver
thurman	pulpFiction
travolta	pulpFiction

DirectedBy	
pulpFiction	tarantino
taxiDriver	scorsese
whiteRibbon	haneke
winterSleep	ceylan

Awarded	
pulpFiction	palmed'Or
taxiDriver	palmed'Or
whiteRibbon	fibresci
winterSleep	palmed'Or

Given these preliminaries, we can now formulate query answering as a decision problem. Note that, we will mostly focus on the special case of this problem, called Boolean query answering, which we will also refer as query evaluation.

Definition 3.4 (query answering, evaluation) Let σ be a relational vocabulary; $\Phi(x_1, \ldots, x_n)$ be a first-order formula over σ; and $\mathcal{D}$ be a database over σ. Then, *query answering* is to decide whether $\mathcal{D} \models \Phi[x_1/a_1, \ldots, x_n/a_n]$ for a given substitution (answer) $[x_1/a_1, \ldots, x_n/a_n]$ to free variables $x_1, \ldots, x_n$. For a Boolean formula Φ, *Boolean query answering*, or simply *query evaluation*, is to decide whether $\mathcal{D} \models \Phi$. $\Diamond$

There exists a plethora of query languages in the literature. Classical database query languages range from the well-known *conjunctive queries* to arbitrary first-order queries, which we briefly introduce.

Definition 3.5 (query languages) A *conjunctive query* over σ is an existentially quantified formula $\exists\vec{x}.\Phi(\vec{x}, \vec{y})$, where $\Phi(\vec{x}, \vec{y})$ is a conjunction of σ-atoms. A *Boolean conjunctive query* over σ is a conjunctive query without free variables. A *union of conjunctive queries* is a disjunction of conjunctive queries. A *union of conjunctive query* is Boolean if it does not contain any free variable. The class of *Boolean unions of conjunctive queries* is denoted as UCQ.

We always focus on Boolean queries throughout this thesis unless explicitly mentioned otherwise. Unions of conjunctive queries are the most common database queries used, in practice; thus, they will also be emphasized in this work. Besides, note that full relational algebra corresponds to the class of first-order formulas (modulo some operators). Therefore, we include fragments of the class of first-order formulas as query languages in our analysis. In particular, we study $\exists$FO, $\forall$FO and FO queries, introduced in Chapter 2, as query languages. Besides, we sometimes use different syntactic forms to represent relational queries, such as CNF or DNF.

We also speak of matches for Boolean queries. Informally, a *match* is an assignment to the variables in the query such that the resulting ground query is satisfied by the database.

Definition 3.6 (match) Let Q be a Boolean query over σ, $\mathcal{D}$ a database over σ and $\mathbf{V}(Q)$ be the set of variables that occur in Q. A mapping $\varphi : \mathbf{V}(Q) \mapsto \mathbf{C}$ is called a *match for the query* Q in $\mathcal{D}$ if $\mathcal{D} \models \varphi(Q)$. $\Diamond$

For existentially quantified queries, it is sufficient to find a single match, to satisfy a given Boolean query evaluation. Conversely, for universally quantified queries, all mappings must result in a match in order to satisfy the query. Let us now briefly illustrate these notions on the database $\mathcal{D}_m$ given in Table 3.1 and on a simple conjunctive query.

Example 3.7 Let us consider again the database $\mathcal{D}_m$ and the non-Boolean query

$$Q_t(x) \coloneqq \exists y\ \mathsf{StarredIn}(x, y) \wedge \mathsf{DirectedBy}(y, \mathsf{tarantino}),$$

which asks for actors that starred in a Tarantino movie. Answers to such queries are tuples from the database. For example, $Q_t(x)$ has two answers in the given database, e.g. $[x/\mathsf{thurman}]$ and $[x/\mathsf{travolta}]$. For each of these answers, there is a match, namely $[y/\mathsf{pulpFiction}]$, for the resulting Boolean query. We focus on *Boolean* variants of these queries. Answers to such queries are either *true* or *false*. For example, the query

$$\exists x, y\ \mathsf{StarredIn}(x, y) \wedge \mathsf{DirectedBy}(y, \mathsf{tarantino}),$$

where all variables are existentially quantified, returns *true* on the given database since there is a match for the query. ◊

Importantly, UCQ denotes a class of queries (in analogy to ∃FO, ∀FO, and FO); thus, strictly speaking, it is not an abbreviation for "unions of conjunctive queries". Nevertheless, we will slightly abuse this notation for unions of conjunctive queries and write "a UCQ Q" instead of "a UCQ query Q".

From a conceptual perspective relational databases can also be viewed as propositional models, where every atom is mapped to a different proposition. Similarly, a database query can be rewritten into a propositional formula by naïvely grounding the query over the database constants and then replacing each ground atom in the resulting formula with a propositional variable. The propositional representation of the query is commonly known as the *lineage* of the query and can be exponentially large in the size of the database.

3.1.2 On The Complexity of Query Evaluation in Databases

When analysing the complexity of query evaluation, we follow Vardi's taxonomy to obtain a fine-grained analysis of the computational complexity (Vardi 1982). The *combined complexity* of query evaluation is calculated by considering all the components, i.e., the database, and the query, as part of the input. The *bounded-arity combined complexity* (or simply bounded-arity complexity) assumes that the maximum arity of the predicates is bounded by an integer constant. The *data complexity* is calculated only based on the size of the database, i.e., the query is assumed to be fixed.

Both data and combined complexity are fairly standard in database theory. Another common approach is to fix the schema, that is, the set of predicates in the signature. We instead only bound the arity of the predicates, which is clearly more general than fixing the schema.

Query evaluation is very efficient in data complexity. As already noted in Chapter 2, the class of FO queries over finite structures can be encoded into AC^0 circuits.

Proposition 3.8 (Immerman 1999) *Query evaluation for* FO *queries is in* AC^0 *in data complexity.*

If we additionally consider the query as part of the input, query evaluation becomes NP-hard even for conjunctive queries.

Proposition 3.9 (Chandra and Merlin 1977) *Query evaluation is* NP*-complete for unions of conjunctive queries and* PSPACE*-complete for* FO *queries in combined complexity.*

It is well-known that this hardness result does not apply to acyclic conjunctive queries, which remain in polynomial time (Yannakakis 1981) and even in AC^1 as shown in (Gottlob, Leone, and Scarcello 2001). Other combined complexity results are directly related to the logical characterization of the polynomial hierarchy.

Proposition 3.10 *Query evaluation is* NP*-complete for* $\exists$FO *queries and* coNP*-complete for* $\forall$FO *queries in combined complexity.*

Furthermore, all the hardness results remain valid for the bounded-arity case. We base our analysis on these known results and extend them to the problems studied in the thesis.

3.2 Probabilistic Databases

Classical databases are *deterministic*: every fact stored in the database is true with absolute certainty, and all the remaining facts are assumed to be false, by the closed-world assumption. As it is the case for classical first-order logic, it is not possible to form statements with intermediate truth values, in classical databases.

On the other hand, there are many *sources of uncertainty* in the real-world, some of which are already highlighted in the general introduction: data is nowadays extracted using automated techniques from the Web, mostly based on data mining or machine learning techniques, which do not necessarily produce a probability distribution, but come with confidence values, and are usually interpreted probabilistically after appropriate transformations are made. Data integration from diverse sources is another source for uncertainty (Dong, Halevy, and Yu 2007). In domains where predictive and stochastic modeling is needed, data usually comes with some probabilistic information. Sensor readings, which are widely used nowadays, are error-prone, and thus are yet another source for uncertainty. Unavoidably, the probabilistic modeling is closely related to the respective application. In this work, we usually abstract away from subtle, practical differences and assume that the data is encoded into a particular probabilistic database model.

The literature on probabilistic databases is sparse and there exists many different types of probabilistic database models. We refer the interested reader to (Suciu et al. 2011) for details of these models and we focus on a particular model.

3.2.1 Tuple-Independent Probabilistic Databases

We adopt the simplest probabilistic database model, which is based on the *tuple-independence assumption*. Syntactically, tuple-independent probabilistic databases generalize classical databases by associating every tuple with a probability value.

Definition 3.11 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 $p = q$. $\Diamond$

Table 3.2: The probabilistic database $\mathcal{P}_m$ represented in terms of database tables. Each row is interpreted as a probabilistic atom $\langle t : p \rangle$, where t is a (ground) atom and p represents its probability.

StarredIn		P
deNiro	taxiDriver	0.7
foster	taxiDriver	0.2
thurman	pulpFiction	0.1
travolta	pulpFiction	0.3

DirectedBy		P
pulpFiction	tarantino	0.8
taxiDriver	scorsese	0.6
whiteRibbon	haneke	0.7
winterSleep	ceylan	0.8

Awarded		P
pulpFiction	palmed'Or	0.2
taxiDriver	palmed'Or	0.9
whiteRibbon	fibresci	0.7
winterSleep	palmed'Or	0.6

Table 3.2 shows an example PDB $\mathcal{P}_m$; as before, each row in a table represents an atom, which is now also associated with a probability value. Semantically, a PDB can be viewed as a factored representation of exponentially many *possible worlds* (classical databases), each of which has a probability to be true. Both in the AI (De Raedt, Kimmig, and Toivonen 2007; Poole 1997; Sato 1995; Sato and Kameya 1997) and database literature (Suciu et al. 2011), this is commonly refered as the *possible world semantics*.

In PDBs, each database atom is viewed as an independent Bernoulli random variable by the *tuple-independence assumption*. Each *world* is then simply a *classical database*, which sets a choice for all database atoms in the PDB. Furthermore, the *closed-world assumption* forces all atoms that are not present in the database, to have probability zero.

Definition 3.12 A PDB $\mathcal{P}$ for vocabulary σ induces a *unique probability distribution* $\mathrm{P}_\mathcal{P}$ over the set of σ-interpretations (possible worlds) $\mathcal{D}$ such that

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

where the probability of each atom is given as

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

Whenever the probabilistic database is clear from the context, we simply write $\mathrm{P}(t)$, instead of $\mathrm{P}_\mathcal{P}(t)$. We say that a database is *induced by* a PDB $\mathcal{P}$ if it is a possible world (with a non-zero probability) of $\mathcal{P}$. $\Diamond$

Observe that the choice of setting $\mathrm{P}_\mathcal{P}(t) = 0$ for tuples missing from PDB $\mathcal{P}$ is a *probabilistic counterpart* of the closed-world assumption. We will revisit this choice in Chapter 4. Let us now illustrate the semantics of PDBs on a simple example.

Example 3.13 Consider again the PDB $\mathcal{P}_m$ given in Table 3.2. Furthermore, we define the database $\mathcal{D}_1$, which consists of only the atoms $\mathsf{StarredIn}(\mathsf{deNiro}, \mathsf{taxiDriver})$, $\mathsf{DirectedBy}(\mathsf{taxiDriver}, \mathsf{scorsese})$, and $\mathsf{Awarded}(\mathsf{taxiDriver}, \mathsf{palmed'Or})$.

The probability of the world $\mathcal{D}_1$ can then be computed by multiplying the probabilities of the atoms that appear in $\mathcal{D}_1$ with the dual probability of the atoms that do not appear in $\mathcal{D}_1$ as follows

$$\begin{aligned} \mathrm{P}(\mathcal{D}_1) = 0.7 \cdot (1-0.2) \cdot (1-0.1) \cdot (1-0.3) \\ (1-0.8) \cdot 0.6 \cdot (1-0.7) \cdot (1-0.8) \\ (1-0.2) \cdot 0.9 \cdot (1-0.7) \cdot (1-0.6). \end{aligned}$$ ◊

Query languages remain the same in PDBs in the syntactic sense, but the queries are now interpreted through the possible world semantics. As we formalize next, this amounts to walking through all the possible worlds and summing over the probabilites of those worlds which satisfy the query.

Definition 3.14 (query semantics) Let Q be a Boolean query and $\mathcal{P}$ be a PDB. The *probability* of Q in the PDB $\mathcal{P}$ is defined as

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

where $\mathcal{D}$ ranges over all possible worlds. ◊

In general, there are exponentially many worlds and in some cases, it is unavoidable to go through all of them in order to compute the probability. This is very infeasible, but as we shall see later, in some cases, computing the query probability is actually easy.

Example 3.15 Consider again the PDB $\mathcal{P}_m$. In order to evaluate the Boolean query

$$Q \coloneqq \exists x, y \; \mathsf{StarredIn}(x, y) \wedge \mathsf{Awarded}(y, \mathsf{palmed'Or})$$

on $\mathcal{P}_m$, we can naïvely check, for each world $\mathcal{D}$, whether $\mathcal{D} \models Q$. One such world is $\mathcal{D}_1$ as it clearly satisfies $\mathcal{D}_1 \models Q$. Afterwards, we only need to sum over the probabilities of the worlds, for which the satisfaction relation holds, in order to obtain the probability of the query. ◊

In the given example, it is *easy* to compute the probability of the query. Notably, this is the case for any PDB and not only for our toy PDB once we focus on the data complexity. The next section is dedicated to give an understanding of *easy* and *hard* queries along with some new results.

3.2.2 On the Complexity of Inference in Probabilistic Databases

In this section, we provide a short overview on existing complexity results for inference in (tuple-independent) Probabilistic Databases including a data complexity dichotomy result. We also present some new combined complexity results. In our analysis, we are interested in the decision problem of probabilistic query evaluation, as defined next.

Definition 3.16 (probabilistic query evaluation) Given a PDB $\mathcal{P}$, a query Q and a threshold value $p \in [0, 1)$, *probabilistic query evaluation*, denoted PQE, is to decide whether $\mathrm{P}_{\mathcal{P}}(Q) > p$. PQE is parametrized with a particular query language; thus, we write PQE(Q) to define PQE on the class of Q queries. ◊

Remark. *Within the scope of this thesis, we make the following assumptions for the complexity study. We always allow the threshold value to depend on the input. Notice that this is a reasonable assumption since the probability computation obviously depends on the data. We also assume that the threshold values as well as the probability values are always given as rational numbers from the interval* $[0, 1]$.

When the probabilistic database is clear from the context, we will simply write $\mathrm{P}(Q)$ in place of $\mathrm{P}_{\mathcal{P}}(Q)$. Besides, we note that there are other possible ways of defining this decision problem. For instance, the comparison operator for the threshold can be $<$, $\leq$, or $\geq$ (instead of $>$). We fix the operator $>$ as it is more common in the AI literature (Darwiche 2009). Moreover, as we focus on PDBs, the event probabilities are always directly encoded in the input. We show that it is always possible to reduce the test for $\geq$ to the test for $>$, and vice versa for certain probability distributions. To show this, we define a rational number

$$\varepsilon = 0.\underbrace{0\ldots0}_{n}1,$$

where n refers to a precision that is strictly more than the precision of any world in the probability distribution. Whenever such a value, called *ε-value*, is computable in a very efficient way, operators $\geq$ and $>$ can be used interchangeably and the same holds for the operators $\leq$ and $<$.

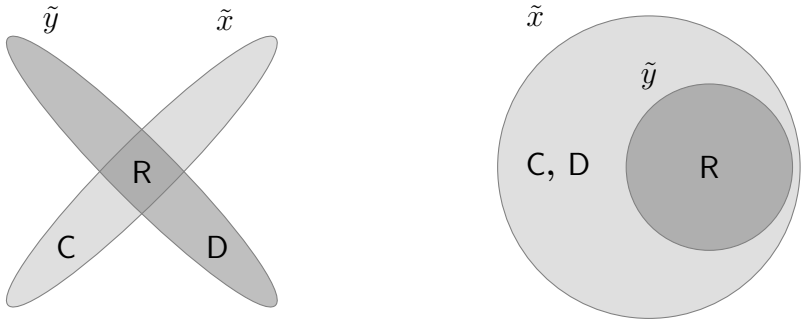
Lemma 3.17 *Let* P *be a probability distribution, where the ε-value can be computed in* AC^0. *Then, for any event θ, deciding* $\mathrm{P}(\theta) > p$ *can be reduced to deciding* $\mathrm{P}(\theta) \geq p+\varepsilon$ *in* AC^0 *and deciding* $\mathrm{P}(\theta) \geq p$ *can be reduced to deciding* $\mathrm{P}(\theta) \geq p-\varepsilon$ *in* AC^0. *Similarly, deciding* $\mathrm{P}(\theta) < p$ *can be reduced to deciding* $\mathrm{P}(\theta) \leq p-\varepsilon$ *in* AC^0 *and deciding* $\mathrm{P}(\theta) \leq p$ *can be reduced to deciding* $\mathrm{P}(\theta) < p+\varepsilon$ *in* AC^0.

Proof. Suppose that an ε-value of the probability of each world can be computed in AC^0. For instance, this is the case for PDBs: the maximal precision of a world in a PDB can be at most $m \cdot n$, where n is the probability value with the highest precision and m is the number of atoms in the PDB. To obtain an ε-value, we can simply shift this number by 1 bit. This requires addition and bit shifting, all of which is in AC^0. Let now θ be an event for which we want to decide whether $\mathrm{P}(\theta) > p$. It is easy to verify that $\mathrm{P}(\theta) > p$ if and only if $\mathrm{P}(\theta) \geq p + \varepsilon$. Conversely, let now θ be an event for which we want to decide whether $\mathrm{P}(\theta) \geq p$. It is easy to verify that $\mathrm{P}(\theta) \geq p$ if and only if $\mathrm{P}(\theta) > p - \varepsilon$. Analogous arguments hold also for $<$ and $\leq$. □

Lemma 3.17 certainly gives us some liberty in the use of the operators $\geq$ and $>$. In the remaining of this text, we will not distinguish between $\geq$ and $>$ in the proof details and will simply use them interchangeably, without further notice.

Overview of Existing Results: A Data Complexity Dichotomy

The data complexity of query evaluation depends heavily on the structure of the query. In a remarkable result, Dalvi and Suciu proved in 2012 the following dichotomy: the probability of a UCQ can be computed either in FP or it is #P-hard on any PDB. Using the terminology from (Dalvi and Suciu 2012), we say that queries are *safe* if the



(a) $Q_{\mathsf{NH}} \coloneqq \exists x, y\, \mathsf{C}(x) \wedge \mathsf{R}(x,y) \wedge \mathsf{D}(y)$ (b) $Q_{\mathsf{H}} \coloneqq \exists x, y\, \mathsf{C}(x) \wedge \mathsf{R}(x,y) \wedge \mathsf{D}(x)$

Figure 3.1: Venn diagram for the queries Q_{NH} (non-hierarchical) and Q_{H} (hierarchical).

computation problem is in FP, and *unsafe*, otherwise. Probabilistic query evaluation, as defined here, is the corresponding decision problem. It is easy to see that this problem is either in P or it is PP-complete as a corollary to the result of Dalvi and Suciu.

Corollary 3.18 (Dalvi and Suciu 2012) $\mathsf{PQE}(\mathsf{UCQ})$ *is either in* P *or it is* PP*-complete for PDBs in data complexity under polynomial-time Turing reductions.*

Proof. Let a UCQ Q be *safe* for PDBs. Then, for any PDB $\mathcal{P}$, the computation problem $\mathrm{P}(Q)$ uses only polynomial time. As a consequence, it is possible to decide whether the probability exceeds a given threshold p in polynomial time. Thus, $\mathsf{PQE}(\mathsf{UCQ})$ is in P for all *safe* UCQ queries.

Conversely, let a UCQ Q be *unsafe* for PDBs. Then, there exists a PDB $\mathcal{P}$ such that computing $\mathrm{P}(Q)$ is #P-hard *under polynomial-time Turing reductions*. Let us loosely denote by $\mathrm{P}(Q)$ the problem of computing $\mathrm{P}(Q)$. We need to show that PP is contained in $\mathrm{P}^{\mathsf{PQE}(Q)}$.

To show this, let A be any other problem in PP. By assumption, its computation problem, denoted $\#\mathsf{A}$, is contained in $\mathrm{FP}^{\mathrm{P}(Q)}$, i.e., there is a polynomial-time Turing machine with oracle $\mathrm{P}(Q)$ that computes the output for $\#\mathsf{A}$. We can adapt this Turing machine then to compare the output to some threshold, which means that A is contained in $\mathrm{P}^{\mathrm{P}(Q)}$. We also know that $\mathrm{P}(Q)$ is contained in $\mathrm{FP}^{\mathsf{PQE}(Q)}$ as we can perform a binary search over the interval $[0,1]$ to compute the precise probability $\mathrm{P}(Q)$. This implies that A is contained in $\mathrm{P}^{\mathfrak{C}}$ where $\mathfrak{C} = \mathrm{FP}^{\mathsf{PQE}(Q)}$. Finally, note that the intermediate oracle does not provide any additional computational power (as this computation can be performed by the polynomial time Turing machine and the oracle $\mathsf{PQE}(Q)$ can be queried directly). This shows that A is in $\mathrm{P}^{\mathsf{PQE}(Q)}$, which proves the result. □

We use the same terminology also for the associated decision problem: we say that a query Q is *safe* if the $\mathsf{PQE}(Q)$ is in P, and *unsafe*, otherwise. (Grädel, Gurevich, and Hirsch 1998) were the first to consider a dichotomy and later, Dalvi and Suciu prove a dichotomy result, which applies to a subclass of conjunctive queries, commonly refered as the *small dichotomy result*. As it gives nice insights on the larger dichotomy result, we briefly look into this, while also taking the opportunity to introduce some intricate notions; all these notions are relevant in the remaining of the thesis. The small dichotomy applies to all conjunctive queries without self-joins, i.e., conjunctive queries

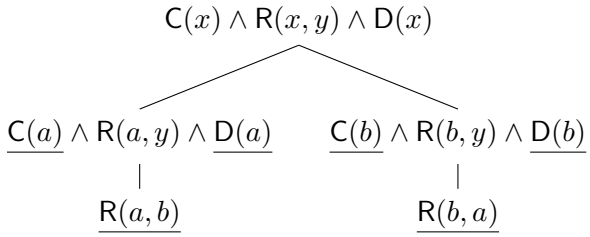


Figure 3.2: Decomposition tree of a safe query for the grounding $[x/a, y/b]$ (left branch) and $[x/b, y/a]$ (right branch). Different branches of the tree do not share an atom, which ensures independence.

with non-repeating relation symbols. It asserts that a self-join free query is hard if and only if it is *nonhierarchical* and it is safe, otherwise. It is therefore crucial to understand *hierarchical* and *nonhierarchical* queries.

Definition 3.19 (hierarchical queries) Let Q be a conjunctive query. For any variable x that appears in the query Q, its *x-cover*, denoted $\tilde{x}$, is defined as the set of all relation names that have the variable x as an argument. Two covers $\tilde{x}$ and $\tilde{y}$ are *pairwise hierarchical* if and only if $\tilde{x} \cap \tilde{y} \neq \emptyset$ implies $\tilde{x} \subseteq \tilde{y}$ or $\tilde{y} \subseteq \tilde{x}$. A query Q is *hierarchical* if every cover $\tilde{x}, \tilde{y}$ is *pairwise hierarchical*; otherwise, it is called *nonhierarchical*. ◊

Let us consider the query $Q_{\mathsf{NH}} \coloneqq \exists x, y\, \mathsf{C}(x) \wedge \mathsf{R}(x, y) \wedge \mathsf{D}(y)$. Observe that this query is not hierarchical since the relation R occurs in both covers $\tilde{x}$ and $\tilde{y}$ (as depicted in Figure 3.1a). As shown in (Dalvi and Suciu 2007), this simple join query is already unsafe. However, removing any of the atoms from this query results in a safe query. For example, the query $\exists x, y\ \mathsf{C}(x) \wedge \mathsf{R}(x, y)$ is hierarchical and thus safe. The query $Q_{\mathsf{H}} \coloneqq \exists x, y\ \mathsf{C}(x) \wedge \mathsf{R}(x, y) \wedge \mathsf{D}(x)$, as shown in Figure 3.1b, is another example of a safe query.

The intuition behind a safe query is the query being recursively decomposable into sub-queries such that each such sub-query is probabilistically independent. Let us consider the query Q_{H} as it admits a decomposition, and is safe. We can first ground over x, which results in a query of the form $\exists y\ \mathsf{C}(a) \wedge \mathsf{R}(a, y) \wedge \mathsf{D}(a)$ for a grounding $[x/a]$. The atoms in the resulting query do not share a relation name or a variable and since we additionally assume tuple-independence, it follows that the probability of each atom is independent. Thus, their probabilities can be computed separately and combined afterwards using appropriate rules of probability.

Note that our observation for the independence is also valid for all different groundings of Q_{H}. For example, the groundings $Q_{\mathsf{H}}[x/a]$ and $Q_{\mathsf{H}}[x/b]$, are probabilistically independent since after applying a grounding over y, we obtain *mutually disjoint sets* of ground atoms. In other words, once x is mapped to different constants, then all mappings for y will result in different sets of atoms. As a result, their probabilities can be computed separately and combined afterwards. The decomposition of the safe query Q_{H} is depicted in Figure 3.2 in terms of a tree. The key ingredient in this example is related to the variable x, which serves as a separator variable in the first place and allows us to further simplify the query.

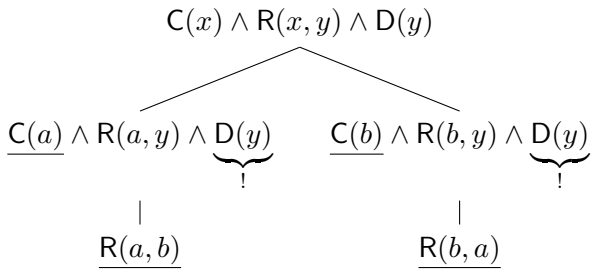


Figure 3.3: Decomposition tree of an unsafe query for the grounding $[x/a, y/b]$ (left branch) and $[x/b, y/a]$ (right branch). Different branches of the tree share D-atoms, which makes them dependent.

Definition 3.20 (separator variable) Let Q be a first-order query. A variable x in Q is a separator variable if x appears in all atoms of Q and for any two different atoms of the same relation R, the variable x occurs in the same position. ◊

Note that the query Q_{NH} has no separator variable since neither x, nor y serve as a separator variable. On an intuitive level, this means that the query can not be decomposed into independent sub-queries. For example, two different groundings $Q_{\mathsf{NH}}[x/a]$ and $Q_{\mathsf{NH}}[x/b]$ are not independent for Q_{NH} since they do not necessarily result in mutually exclusive sets of atoms once grounded over y as shown in Figure 3.3.

The small dichotomy theorem uses other rules of probability theory to further simplify the query; we refer to (Suciu et al. 2011), for further details. The dichotomy for unions of conjunctive queries is much more intricate and a characterization of safe queries is unfortunately not easy. Thus, Dalvi and Suciu define an algorithm that computes the probability of all safe queries by recursively applying the simplification rules on the query. This algorithm is complete, i.e., when the algorithm fails on the query, then the query is unsafe. Later, a lifted inference algorithm, called $\mathsf{Lift}^{\mathsf{R}}$, was proposed in (Gribkoff, Van den Broeck, and Suciu 2014b), which was also proven to be complete. For details, we refer to the relevant literature and defer some details to Chapter 4, where we lift the dichotomy for unions of conjunctive queries to an open-world semantics. Note that this is achieved by a modified lifted inference algorithm, called $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$, which properly extends the existing algorithm $\mathsf{Lift}^{\mathsf{R}}$.

Safety is thus a property that can be completely determined for unions of conjunctive queries in polynomial time in data complexity (based on a complete algorithm). A natural question that could arise is whether a similar classification would be possible for arbitrary first-order queries. Unfortunately, safety is not a decidable property for the class of FO queries as noted in (Suciu et al. 2011). It is a simple consequence of the undecidability of finite satisfiability (Trakhtenbrot 1950). Consider the unsafe UCQ Q_{NH} and an arbitrary FO query Q that does not share any relation names with Q_{NH}. It is then easy to see that $Q_{\mathsf{NH}} \wedge Q$ is safe if and only if Q is unsatisfiable.

Proposition 3.21 (Suciu et al. 2011) *Safety is undecidable for* FO *queries.*

Clearly, this negative result does not mean that first-order queries can not be evaluated over PDBs. Leaving aside the classification of the queries, we can show that PQE(FO) is

PP-complete. Importantly, the hardness result holds even for ∃FO and ∀FO queries. At first sight, this PP-hardness result may seem obvious since PP-hardness already holds for unions of conjunctive queries as a result of the dichotomy. Recall, however, that the dichotomy is shown under polynomial time Turing reductions. It is *open* whether there exists a UCQ query, for which probabilistic query evaluation is PP-hard under polynomial time many-one reductions. Our hardness result implies that this is the case for ∃FO or ∀FO queries.

Theorem 3.22 PQE(∃FO), PQE(∀FO), PQE(FO) *are* PP*-complete for PDBs in data complexity.*

Proof. Let $\mathcal{P}$ be a PDB, Q a FO query and $p \in (0, 1]$ a threshold value. Let us denote by P the probability distribution induced by $\mathcal{P}$. There are exponentially many databases (worlds) $\mathcal{D}$ induced by $\mathcal{P}$, each of which holds with some probability. We now create multiple copies of each world in such a way that the uniform distribution over all thus generated worlds is equivalent to P when each copy is taken to represent its original world. Given this uniform distribution over the worlds, we now consider a nondeterministic Turing machine, where each branch corresponds to one of these worlds. Each branch of the nondeterministic Turing machine represents an accepting run if the test $\mathcal{D} \models Q$ is positive for the corresponding world $\mathcal{D}$ (which can be verified in polynomial time in data complexity). Moreover, for threshold values properly above (respectively, below) 0.5, we introduce artificial success (respectively, failure) branches into the nondeterministic Turing machine such that satisfying the original threshold corresponds to having a majority of successful computations. Then, the answer to the probabilistic query entailment problem is *yes* if and only if the nondeterministic Turing machine answers *yes* in the majority of its runs, which proves membership.

As for the lower bound, we first prove the result for ∀FO queries by reducing the following problem. Let $\Phi \coloneqq \mathsf{C}^c\, x_1, \ldots, x_n\ \varphi$ denote a quantified Boolean formula, where C represents the *counting quantifier* and $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_k$ is a propositional formula in 3CNF, defined over the variables $x_1, \ldots, x_n$. Deciding the validity of such formulas is PP-complete (Wagner 1986). Intuitively, this amounts to checking whether there are c assignments for $x_1, \ldots, x_n$ that satisfy φ. For the reduction, we consider the following ∀FO query

$$
\begin{aligned}
Q_{\mathsf{SAT}} \coloneqq \forall x, y, z\ (\ & \mathsf{L}(x) \vee \ \mathsf{L}(y) \vee \ \mathsf{L}(z) \vee \mathsf{R}_1(x, y, z)) \wedge \\
& (\neg\mathsf{L}(x) \vee \ \mathsf{L}(y) \vee \ \mathsf{L}(z) \vee \mathsf{R}_2(x, y, z)) \wedge \\
& (\neg\mathsf{L}(x) \vee \neg\mathsf{L}(y) \vee \ \mathsf{L}(z) \vee \mathsf{R}_3(x, y, z)) \wedge \\
& (\neg\mathsf{L}(x) \vee \neg\mathsf{L}(y) \vee \neg\mathsf{L}(z) \vee \mathsf{R}_4(x, y, z))\ ,
\end{aligned}
$$

which is used to encode the satisfaction conditions of the formula Φ. Furthermore, we define the PDB $\mathcal{P}_\Phi$ that stores the structure of Φ as follows.

- For each variable x_i, $1 \leq i \leq n$, $\mathcal{P}_\Phi$ contains the atoms $\langle \mathsf{L}(x_i) : 0.5 \rangle$, where we view each x_i as a database constant.
- The clauses φ_i are described with the help of the predicates R_1, …, R_4, each of which corresponds to one type of clause. For example, if we have the clause $\varphi_i = x_1 \vee \neg x_2 \vee \neg x_4$, we add the atom $\langle \mathsf{R}_3(x_4, x_2, x_1) : 0 \rangle$ to $\mathcal{P}_\Phi$, which enforces

via Q_{SAT} that either $\neg\mathsf{L}(x_4)$, $\neg\mathsf{L}(x_2)$ or $\mathsf{L}(x_1)$ holds. All other R-atoms that do not correspond in such a way to one of the clauses, we add with probability 1 to $\mathcal{P}_\Phi$.

Claim. The formula Φ is valid if and only if $\mathrm{P}_{\mathcal{P}_\Phi}(Q_{\mathsf{SAT}}) \geq \mathsf{c} \cdot (0.5)^n$

Suppose that Φ is valid. Then, there are at least c different assignments τ to the variables $x_1, \ldots, x_n$ that satisfy φ. For each satisfying assignment τ, we define a database $\mathcal{D}$ that contains an atom $\mathsf{L}(x_i)$ if and only if τ sets x_i to true in Φ. Moreover, we add all R-atoms that are in $\mathcal{P}_\Phi$ with probability 1. It is easy to see that each such database $\mathcal{D}$ is a world induced by $\mathcal{P}_\Phi$ and satisfies $\mathcal{D} \models Q_{\mathsf{SAT}}$ (as each such world is in one-to-one correspondence with a satisfying valuation). Finally, it suffices to observe that there are only n probabilistic atoms in $\mathcal{P}_\Phi$; namely the atoms $\mathsf{L}(x_j)$, $1 \leq i \leq n$, that correspond to the variables in Φ. Thus, every database $\mathcal{D}$ induced by $\mathcal{P}_\Phi$ has the probability 0.5^n. By our assumption, there are c satisfying assignments τ to Φ; thus, it follows that $\mathrm{P}_{\mathcal{P}_\Phi}(Q_{\mathsf{SAT}}) \geq \mathsf{c} \cdot (0.5)^n$.

For the other direction, let $\mathrm{P}_{\mathcal{P}_\Phi}(Q_{\mathsf{SAT}}) \geq \mathsf{c} \cdot (0.5)^n$. Then, each database $\mathcal{D}$ induced by $\mathcal{P}_\Phi$ sets a choice for the nondeterministic atoms $\mathsf{L}(x_1), \ldots, \mathsf{L}(x_n)$ and each such database has the probability $(0.5)^n$ (as there are only n nondeterministic atoms in the PDB). As a consequence, there must exist at least c databases induced by $\mathcal{P}_\Phi$ that satisfies $\mathcal{D} \models Q$.

For each such database $\mathcal{D}$, we define a corresponding assignment τ to the variables $x_1, \ldots, x_n$ such that x_i is mapped to true in τ if and only if $\mathsf{L}(x_i) \in \mathcal{D}$. It is then easy to verify that $\tau \models \varphi$. As there are c different assignments τ that satisfy φ, we conclude that the formula Φ is valid. This proves PP-hardness for $\forall\mathsf{FO}$ queries.

Finally, observe that the negation of Q_{SAT} is an $\exists\mathsf{FO}$ query and PP is closed under complement as it is closed under truth table reductions (Beigel, Reingold, and Spielman 1995). Thus, this hardness also holds for $\exists\mathsf{FO}$ queries. □

Notice that the query in the given reduction uses the power of negation, which makes the many-one reduction possible. It remains *open* whether there exists a positive query, for which probabilistic query evaluation is PP-hard under many-one reductions. We now continue our analysis with combined complexity results.

Beyond Existing Results: The Combined Complexity

In the context of (probabilistic) databases, the study of combined complexity seems to be mostly ignored in the literature. This is mainly due to the fact that data complexity often captures the real-world complexity of the relevant problems in a more adequate manner. On the other hand, it is not hard to imagine scenarios where a safe query (in data complexity) could require super-polynomial time in the query. Similar observations motivated some work on this subject; for instance, there is recent work to isolate cases where PQE is tractable in combined complexity (Amarilli, Monet, and Senellart 2017) .

There is yet another subtle reason for (mostly) abandoning combined complexity analysis in PDBs, which is of a very technical nature: all existing data complexity results are shown under Turing reductions, which leads to the collapse of many interesting classes, which could possibly make a difference in the case of combined complexity. Our combined complexity analysis, as any other result in this work except from dichotomy results, are under many-one reductions, and therefore, we obtain more fine-grained characterizations.

We first show that probabilistic query evaluation for unions of conjunctive queries is complete for $\mathrm{PP}^{\mathrm{NP}}$ in combined complexity.

Theorem 3.23 $\mathsf{PQE}(\mathsf{UCQ})$ *is* $\mathrm{PP}^{\mathrm{NP}}$*-complete for PDBs in combined complexity.*

Proof. Let $\mathcal{P}$ be a PDB, Q a UCQ and $p \in (0, 1]$ a threshold value. As before, we define a nondeterministic Turing machine as in the proof of Theorem 3.22. Differently, the test $\mathcal{D} \models Q$ is NP-complete in combined complexity (and it remains in NP for $\exists\mathsf{FO}$ queries). The answer to this test can be retrieved from the oracle machine, which proves membership.

In order to show hardness, we reduce the following problem: decide validity of formulas of the form

$$\Phi \coloneqq \mathsf{C}^c\, x_1, \ldots, x_m \exists y_1, \ldots, y_n\ \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k\,,$$

where every φ_i is a propositional clause over $x_1, \ldots, x_m$, $y_1, \ldots, y_n$, and $k, m, n \geq 1$. Φ is valid if and only if, for at least c of the partial assignments μ to $x_1, \ldots, x_m$, the formula

$$\exists y_1, \ldots, y_n\, \mu(\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k),$$

is true. This is a $\mathrm{PP}^{\mathrm{NP}}$-complete problem (Wagner 1986). To simplify the proof, we also assume, without loss of generality, that φ contains all clauses of the form $x_j \vee \neg x_j$, $1 \leq j \leq m$, and similarly $y_j \vee \neg y_j$, $1 \leq j \leq n$; clearly, this does not affect the existence or number of satisfying assignments for φ. We also assume that each clause φ_j contains exactly three literals. This is also without loss of generality, since otherwise we can introduce additional existentially quantified variables to abbreviate the clauses, or duplicate literals if the clauses are too short. We define the PDB $\mathcal{P}_\Phi$ for the reduction as follows.

- For each variable x_j, $1 \leq j \leq m$, it contains the atoms $\langle \mathsf{L}(x_j, 0) : 0.5 \rangle$ and $\langle \mathsf{L}(x_j, 1) : 0.5 \rangle$.
- Each clause φ_j is described with the help of a predicate $\mathsf{M}(\cdot, \cdot, \cdot, j)$ of arity 4, which encodes the satisfying assignments for φ_j. For example, consider the clause $\varphi_j = x_2 \vee \neg y_4 \vee y_1$. For the satisfying assignment $x_2 \mapsto \mathit{true}$, $y_4 \mapsto \mathit{true}$, $y_1 \mapsto \mathit{false}$, we add the atom $\mathsf{M}(1, 1, 0, j)$ with probability 1, and similarly for all other satisfying assignments. There are at most 7 satisfying assignments for each clause.

Furthermore, we define the UCQ

$$Q_\Phi \coloneqq (\exists y_1, \ldots, y_n\, \psi_1 \wedge \cdots \wedge \psi_k) \vee (\exists x\, \mathsf{L}(x, 0) \wedge \mathsf{L}(x, 1)),$$

where each ψ_j is a conjunction that is derived from φ_j depending on the types of the involved variables. We describe the details again on the example clause $\varphi_j = x_2 \vee \neg y_4 \vee y_1$. The satisfaction of this clause is encoded by the conjunction

$$\psi_j = \mathsf{M}(i, y_4, y_1, j) \wedge \mathsf{L}(x_2, i),$$

where i is an additional existentially quantified variable that is local to ψ_j, and j is fixed. Intuitively, ψ_j asserts that the truth assignment for x_2, y_4, and y_1 (given by x_2, i, and y_1, respectively) satisfies φ_j. Note that the variables $y_1, \ldots, y_n$ have to be mapped to 0 or 1, since otherwise they cannot satisfy the M-atoms. Moreover, observe that an alternative

way of satisfying Q_Φ is due to the last clause in Q_Φ: it applies when L-atoms represent an inconsistent assignment (in Φ) for at least one variable of the form x_j. Note that, in this case, the query can be satisfied without actually satisfying the original formula Φ. This happens only if the PDB contains both $\langle \mathsf{L}(x_j, i_1) : 0.5\rangle$, $\langle \mathsf{L}(x_j, i_2) : 0.5\rangle$ for different i_1 and i_2. As there are $2m$ nondeterministic atoms in $\mathcal{P}_\Phi$, there are 4^m worlds; among them, $(4^m - 3^m)$ satisfy the last clause of the query Q_Φ, which corresponds to an inconsistent valuation in Φ. Based on the given construction, and these observations, we now prove the following claim.

Claim. Φ is valid if and only if $\mathrm{P}_{\mathcal{P}_\Phi}(Q_\Phi) \geq 0.5^{2m}(4^m - 3^m + \mathsf{c})$.

Suppose that Φ is valid. Then, there are at least c assignments μ for $x_1, \ldots, x_m$ such that each of these assignments admit an extension τ to the variables $y_1, \ldots, y_n$ such that $\tau \models \varphi$. For each partial valuation μ, we define a database $\mathcal{D}_\mu$ such that it contains all atoms from $\mathcal{P}_\Phi$ that occur with probability 1. Moreover, $\mathcal{D}_\mu$ contains an atom $\mathsf{L}(x_j, 1)$ if x_j is mapped to true in μ, and an atom $\mathsf{L}(x_j, 0)$ if x_j is mapped to false in μ. It is easy to see that each such database $\mathcal{D}_\mu$ is induced by the PDB $\mathcal{P}_\Phi$. Besides, since each of these assignments μ admit an extension τ to the variables $y_1, \ldots, y_n$ such that $\tau \models \varphi$, it follows that $\mathcal{D}_\mu \models (\exists y_1, \ldots, y_n\, \psi_1 \wedge \cdots \wedge \psi_k)$, as all satisfying assignments are already encoded in the database. In particular, this implies that $\mathcal{D}_\mu \models Q_\Phi$ for c worlds. Recall also that $(4^m - 3^m)$ worlds satisfy the last clause in the query (which captures the inconsistent valuations). As every world has the probability $(0.5)^{2m}$, we conclude that $\mathrm{P}_{\mathcal{P}_\Phi}(Q_\Phi) \geq 0.5^{2m}(4^m - 3^m + c)$.

Conversely, if $\mathrm{P}_{\mathcal{P}_\Phi}(Q_\Phi) \geq 0.5^{2m}(4^m - 3^m + c)$, then there are at least c worlds that satisfy the first clause in Q_Φ. For each of those worlds $\mathcal{D}$, we define a partial assignment $\mu_\mathcal{D}$ such that a variable x_j is mapped to true if $\mathsf{L}(x_j, 1) \in \mathcal{D}$ and it is mapped to false if $\mathsf{L}(x_j, 0)$. Moreover, $\mathcal{D} \models (\exists y_1, \ldots, y_n\, \psi_1 \wedge \cdots \wedge \psi_k)$ implies that there is a satisfying mapping for the y-variables in the database. Recall that, this can only be the case if a variable y_j is either mapped to 0 or to 1 due to the structure encoded in M-atoms. We define an extension $\tau_\mathcal{D}$ of $\mu_\mathcal{D}$, which maps a variable y_j to true if and only if it is mapped to 1 in the database and to false, otherwise. It is easy to verify that $\tau_\mathcal{D} \models \varphi$. Thus, for c partial assignments, the formula $\exists y_1, \ldots, y_n\, \varphi_1 \wedge \cdots \wedge \varphi_k$ is satisfiable; meaning that, the formula Φ must be valid. □

Importantly, in the hardness proof of Theorem 3.23, all the predicates are of a bounded arity; that is, the proof already applies to the *bounded-arity* combined complexity and so do the remaining results in this section.

Theorem 3.24 $\mathsf{PQE}(\exists\mathsf{FO})$ *and* $\mathsf{PQE}(\forall\mathsf{FO})$ *is* $\mathrm{PP}^{\mathrm{NP}}$*-complete for PDBs in combined complexity.*

Proof. The lower bounds follow from Theorem 3.23 and similarly the membership for $\exists\mathsf{FO}$ queries. For universal queries, the only difference is that the verification step $\mathcal{D} \models Q$, that needs to be performed for each world $\mathcal{D}$, is coNP-complete in combined complexity. This implies a $\mathrm{PP}^{\mathrm{coNP}}$ upper bound, which is equivalent to $\mathrm{PP}^{\mathrm{NP}}$. □

Query evaluation is already PSpace-complete in databases for FO queries. Therefore, we immediately obtain the following result.

Table 3.3: Data, bounded-arity and combined complexity results for probabilistic query evaluation in PDBs. The data complexity dichotomy is by Dalvi and Suciu and holds under polynomial time Turing reductions. All the remaining results are original contributions and hold under polynomial time many-one reductions.

Query Languages	**Probabilistic Query Evaluation in PDBs**		
	data	bounded-arity	combined
UCQ	P vs PP [Corollary 3.18]	PP^{NP} [Theorem 3.23]	PP^{NP} [Theorem 3.23]
∃FO, ∀FO	PP [Theorem 3.22]	PP^{NP} [Theorem 3.24]	PP^{NP} [Theorem 3.24]
FO	PP [Theorem 3.22]	PSPACE [Theorem 3.25]	PSPACE [Theorem 3.25]

Theorem 3.25 PQE(FO) *is* PSPACE*-complete for PDBs in combined complexity.*

Proof. Let $\mathcal{P}$ be a PDB, Q a FO query and $p \in [0, 1)$, a threshold value. Consider a *polynomial-space bounded* nondeterministic Turing machine that enumerates all (exponentially many) worlds $\mathcal{D}$, and performs the test $\mathcal{D} \models Q$ for those worlds; then, adds up their probabilities if the test is successful. Finally, the machine answers *yes* if and only if the resulting probability exceeds the given threshold, which proves membership.

Hardness is immediate since query evaluation problem in databases for FO queries is already PSPACE-hard in combined complexity (even if we assume that the arity of the predicates are bounded). To simulate this problem, we define a PDB $\mathcal{P}$, which contains all the atoms from a given arbitrary $\mathcal{D}$ with probability 1. Then, for any FO query Q, we have that $\mathcal{D} \models Q$ if and only if $\text{P}_{\mathcal{P}}(Q) \geq 1$. □

All the results given in this section are summarized in Table 3.3. All the results except the data complexity dichotomy (a corollary to the result of (Dalvi and Suciu 2012)) are original contributions. We later show that this dichotomy can be further strengthened (by employing some reasonable assumptions) and all the other results provide a baseline for remaining analysis.

3.2.3 Relations to Weighted Model Counting

Weighted model counting (WMC) is a natural problem that extends model counting (MC) by associating a weight to every model. Our interest stems from the fact that probabilistic query evaluation on a probabilistic database can be reduced to WMC, and, therefore, the theory for WMC can be deployed to perform query evaluation.

In the scope of this thesis, we are concerned with first-order model counting (FOMC) and weighted first-order model counting (WFOMC), which generalize MC and WMC of propositional models. We give a brief background on these problems and describe the connection to query evaluation on probabilistic databases.

Definition 3.26 (WFOMC) Given a first-order formula Φ over a vocabulary σ with a fixed, finite domain, let Γ be the set of σ-literals and $w : \Gamma \mapsto \mathbb{R}$ be a weight function. The *weighted first-order model count* of Φ is defined as

$$\mathsf{WFOMC}(\Phi, w) \coloneqq \sum_{\mathcal{I} \models \Phi} \prod_{\mathcal{I} \models l} w(l) \ ,$$

where $\mathcal{I}$ is a first-order interpretation over the fixed domain and l ranges over the literals in Γ. The *first-order model count* of Φ, denoted $\mathsf{FOMC}(\Phi)$ is the special case of WFOMC where $w(l) = 1$ for all literals $l \in \Gamma$. The *symmetric first-order model count* of Φ, denoted $\mathsf{SWFOMC}(\Phi, w)$, is the special case of WFOMC where for any two literals l and l' belonging to the same predicate, it holds that $w(l) = w(l')$. $\lozenge$

Propositional variants of these problems are defined in the exact same way by replacing the first-order formula with a propositional one and first-order literals with propositional literals. Clearly, there is no notion of symmetric model counting in the propositional setting.

The connection to probabilistic query evaluation in probabilistic databases is then immediate: given a PDB $\mathcal{P}$ and a Boolean query Q

$$\mathrm{P}_{\mathcal{P}}(Q) = \mathsf{WFOMC}(Q, w),$$

where the weight function w is defined such that for all probabilistic atoms $\langle l : p \rangle \in \mathcal{P}$, we set $w(l) = p$ and $w(\neg l) = 1 - p$ whereas for all σ-atoms t that do not appear in $\mathcal{P}$, we set $w(t) = 0$ and $w(\neg t) = 1$.

Notice that a similar correspondence can be achieved with WMC and using the lineage representation of the query relative to the probabilistic database. More formally, let φ be the lineage of Q in the given PDB then

$$\mathrm{P}(\varphi) = \mathsf{WMC}(\varphi, w),$$

where the weight function w is defined as before, with the obvious difference that it now ranges over propositional literals.

While the lineage representation is relatively common, grounding is usually not very efficient, as discussed. Furthermore, it is hard to capture inherent symmetries that may be present in the data using the lineage representation. Performing inference directly on first-order structures is typically more efficient. The differences in the computational complexity of FOMC, WFOMC and SWFOMC give nice insights on this issue: Recall the query $\mathsf{C}(x) \wedge \mathsf{R}(x, y) \wedge \mathsf{D}(y)$, which is unsafe over PDBs in data complexity. Interestingly, the (unweighted) model count of this query can still be computed in polynomial time and remains so if we allow weights that are symmetric, see (Beame, Van den Broeck, Suciu, and Gribkoff 2015), for details and complexity-theoretic assumptions.

On the other hand, first-order model counting comes with its own problems. One obvious issue is the domination problem: predicates with higher arity naturally contain more groundings than those of with lower arity. As a consequence, the total model count tends to be dominated by the higher arity predicates. Although obvious, this problem appears to be overlooked to the best of our knowledge. One way of tackling this problem can be to define domain and range restrictions so that the groundings do not grow artificially in the size of the domain.

3.3 State of the Art in Probabilistic Databases

Historical Background, Semantics and Relations to AI

Research on probabilistic databases is almost as old as traditional databases as stated in (Suciu et al. 2011). Our focus is only on the closely related approaches; for an extended overview, we refer the interested reader to the relevant literature. We note the seminal work of (Fuhr and Rölleke 1997), which has been very important in probabilistic database research as well as the first formulation of possible world semantics in the context of databases by (Imieliński and Lipski 1984).

Possible world semantics has its roots in philosophy and logic and is formalized by (Hintikka 1969) and (Kripke 1963). In artificial intelligence, it has been widely employed in PGMs (Darwiche 2009; Koller and Friedman 2009), which dates back to Bayesian Networks (Pearl 1988). Early applications of possible world semantics in first-order representations can be found in Sato's distribution semantics (Sato 1995) and in Poole's independent choice logic (Poole 1997). Over time, it also became the standard semantics for probabilistic logic programming (De Raedt, Kimmig, and Toivonen 2007) and similarly for probabilistic formal verification (Sato and Kameya 1997).

There is a subtle, but important, issue to note regarding the possible world semantics. Observe that it only defines the method of extending the classical semantics of the underlying formalism (i.e., database, logic program, et cetera) to a probabilistic setting. Hence, all these extended formalisms still have their differences in the semantics as a natural consequence of the differences of the underlying formalisms.

Dichotomy-Related Results

Grädel, Gurevich, and Hirsch were the first to consider a dichotomy and they have shown, for instance that the simple query $\mathsf{C}(x) \wedge \mathsf{R}(x, y) \wedge \mathsf{C}(y)$ is #P-hard. Later, Dalvi and Suciu proved the *small dichotomy result*. The first support for first-order queries in probabilistic databases is in (Fink, Olteanu, and Rath 2011). Moreover, a small dichotomy result for queries with negation has been obtained in (Fink and Olteanu 2016). Other dichotomy results extend the dichotomy for unions of conjuncive queries in other directions; some allow for disequality ($\neq$) joins in the queries (Olteanu and Huang 2008) and some for inequality ($<$) joins in the queries (Olteanu and Huang 2009). There is also a trichotomy result over queries with aggregation (Ré and Suciu 2009). The common ground in all these dichotomy results is the fact that they classify queries as being safe or unsafe (while the data is not fixed). A different approach is to obtain a classification relative to the structure of the underlying database and it has been proven in (Amarilli, Bourhis, and Senellart 2016), for instance, that every query formulated in Monadic Second Order logic can be evaluated in linear time over PDBs with a bounded tree-width.

Inference Techniques

Query answering over PDBs translates into weighted model counting (Gribkoff, Suciu, and Van den Broeck 2014). Most of the existing approaches to model counting are based on propositional models (Chakraborty, Fried, Meel, and Vardi 2015; Chavira

and Darwiche 2008; Gomes, Sabharwal, and B. Selman 2006) and are linked to PDBs through lineage representations. More promising approaches are based on weighted first-order model counting, which exploit inherent symmetries without the explicit need of grounding to a lineage representation (Beame, Li, Roy, and Suciu 2017; Gribkoff, Van den Broeck, and Suciu 2014b).

In general, probabilistic inference is computationally a very demanding task. The most common approach in PGMs as well as in PDBs is to use some approximate algorithms, mostly based on Markov Chain Monte Carlo (MCMC) algorithms; see for instance (Niu, Ré, Doan, and Shavlik 2011) and (Shin et al. 2015). Unfortunately, these algorithms do not provide any guarantees and therefore are less desirable from a theoretical perspective.

Another prominent approach is based on knowledge compilation (Cadoli and Donini 1997; Darwiche and Marquis 2002; B. Selman and Kautz 1996), which is an important branch of knowledge representation and AI that aims at solving difficult AI problems such as *satisfiability*, *validity*, *implication*, and *substitution* in an efficient manner, by compiling the problem into a target language in which it can be solved efficiently: the computational overhead is pushed into an *off-line* preprocessing phase, which is amortised over a large number of *on-line* queries. There are many target languages used in knowledge compilation, ranging from traditional ones such as binary decision diagrams (BDDs), ordered BDDs (OBDDs), and negation normal forms (NNFs) to more sophisticated formalisms like deterministic decomposable NNFs (d-DNNFs), free BDDs (FBDDs), and sentential decision diagrams (SDDs). Different compilation algorithms can solve efficiently different classes of problems, in time polynomial in the size of compiled expression (Darwiche and Marquis 2002).

Recent decades witnessed a shift in AI research towards probabilistic models, and knowledge compilation has been adopted accordingly towards addressing hard probabilistic inference tasks such as *model counting*, or *threshold probability computing*, where typically the former is #P-hard, and the latter PP-hard. Knowledge compilation is widely employed on propositional models (Darwiche and Marquis 2002) and there are already solvers designed for decision problems that belong to complexity classes such as PP and beyond.

Knowledge compilation has also been effective on first-order models (Van den Broeck, Taghipour, Meert, Davis, and Raedt 2011). It is also gaining growing attention in PDBs (Jha and Suciu 2013). A different form of knowledge compilation in PDBs is based on factorising PDBs and queries (Olteanu and Schleich 2016). In particular, the notion of *tensor factorization* characterizes very promising knowledge compilation techniques in the broader sense, which have, e.g., been applied to knowledge base completion (Socher et al. 2013), i.e., the problem of predicting non-existing from existing facts in a knowledge base. Tensor factorization has recently also been applied to PDBs (Krompaß, Nickel, and Tresp 2014).

Systems

Most probabilistic relational database management systems, such as MystiQ (Boulos, Dalvi, Mandhani, Mathur, Ré, and Suciu 2005) and SPROUT (Fink, Hogue, Olteanu, and Rath 2011), are based on the tuple-independence assumption, i.e., any two facts in the database are assumed to be probabilistically independent. In parallel, the crucial

need to relax this independence assumption has already been recognized in several recent approaches, which are based on Markov logic networks (MLNs) (Gribkoff and Suciu 2016b). Here, the PDB is viewed as a set of weighted facts in an MLN; additional soft/hard constraints are imposed through a set of weighted/unweighted rules from MLNs. In particular, the recent system SlimShot (Gribkoff and Suciu 2016b) reduces such PDBs to tuple-independent PDBs with additional logical constraints; it also provides certain accuracy guarantees. Other related approaches based on Markov logic networks, like Tuffy (Niu et al. 2011) and DeepDive (Shin et al. 2015), use MCMC for probabilistic inference, or a variant called MC-SAT (Poon and Domingos 2006). In general, such approximating algorithms do not provide any accuracy guarantees as already pointed out (and also have computational efficiency problems; cf. (Gribkoff and Suciu 2016b)).

Chapter 4

Open-World Probabilistic Databases

We revisit 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, 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. Second, 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. The CWA does not accurately represent this mode of operation and puts it on weak footing. Third, the CWA is problematic for higher level tasks that one is usually interested in performing on probabilistic databases, including some principled approaches to knowledge base completion and mining. Finally, we note that 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 itself 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.

We propose an alternative semantics for probabilistic knowledge bases to address these problems, based on the *open-world assumption (OWA)*. As opposed to the CWA, the OWA does not presume that the knowledge of a domain is complete. Hence, anything that is not in the knowledge base 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 that 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.

4.1 Problems in Probabilistic Databases

We now take a critical view on PDBs and illustrate certain limitations, which are inherent to the semantics of PDBs and the widely employed assumptions. The CWA presumes a complete knowledge about the domain being represented, and this assumption is warranted in many cases (Reiter 1978). For example, *when a flight does not appear in*

Table 4.1: The probabilistic database $\mathcal{P}_c$ represented in terms of database tables.

StarredIn		P
willsmith	ali	0.9
willsmith	sharktale	0.8
jadasmith	ali	0.6
arquette	scream	0.7
pitt	mrmssmith	0.5
jolie	mrmssmith	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

an airline database, we can be sure that *it never took place*. In what follows, we assess the adequacy of the CWA for probabilistic knowledge bases such as NELL, DeepDive, and Knowledge Vault.

4.1.1 Distinguishing Queries

The fact that many queries evaluate to the probability zero makes it impossible to distinguish a large class of queries, which should *intuitively* differ. We will briefly illustrate this effect by considering several queries on the PDB $\mathcal{P}_c$ of Table 4.1.

Example 4.1 (specificity) Consider the PDB $\mathcal{P}_c$ represented as probabilistic database tables in Table 4.1 and the following queries

$$Q_1(x, y) \coloneqq \exists z\, \mathsf{StarredIn}(x, z) \wedge \mathsf{StarredIn}(y, z) \wedge \mathsf{Couple}(x, y),$$
$$Q_2 \coloneqq \exists x, y, z\, \mathsf{StarredIn}(x, z) \wedge \mathsf{StarredIn}(y, z) \wedge \mathsf{Couple}(x, y).$$

Let us consider the queries $Q_1(\mathsf{pitt}, \mathsf{jolie})$ and Q_2. From a logical perspective, $Q_1(\mathsf{pitt}, \mathsf{jolie})$ entails Q_2, i.e., $Q_1(\mathsf{pitt}, \mathsf{jolie}) \models Q_2$. In other words, the pattern specified by $Q_1(\mathsf{pitt}, \mathsf{jolie})$ is only a special case of the pattern specified by Q_2. As a consequence, the reasonable expectation in an open-world setting is that $\mathrm{P}(Q_2) > \mathrm{P}(Q_1(\mathsf{pitt}, \mathsf{jolie}))$, since there exist a large number of couples for which we do not yet have information, could satisfy the query Q_2. Under the CWA, however, $\mathrm{P}(Q_2) = \mathrm{P}(Q_1(\mathsf{pitt}, \mathsf{jolie})) = 0.28$ in the PDB $\mathcal{P}_c$.$\Diamond$

Example 4.1 shows that query semantics under the CWA fails to distinguish a *query* from a particular *instance of this query*. Our next example considers two logically *incomparable queries* that have varying level of *support* in the database.

Example 4.2 (support) Consider now the following queries $Q_1(\mathsf{willsmith}, \mathsf{jadasmith})$ and $Q_1(\mathsf{thornton}, \mathsf{aniston})$. The former query is *supported* by two facts in the PDB $\mathcal{P}_c$, while the latter query is supported by none, which should make it less likely. Conversely, the number of tuples to be added to the PDB $\mathcal{P}_c$ to satisfy $Q_1(\mathsf{thornton}, \mathsf{aniston})$ are more than the number of tuples to be added to the PDB $\mathcal{P}_c$ to satisfy $Q_1(\mathsf{willsmith}, \mathsf{jadasmith})$. Observe, however, that

$$\mathrm{P}(Q_1(\mathsf{thornton}, \mathsf{aniston})) = \mathrm{P}(Q_1(\mathsf{willsmith}, \mathsf{jadasmith})) = 0,$$

i.e., both of the queries evaluate to zero under the CWA. $\Diamond$

Example 4.2 shows that queries that do not have a matching answer in the database are viewed to be the same by the query semantics, even though these queries clearly have different levels of support in the database. The following example takes these observations to the *extreme*, by comparing the probabilities of a *satisfiable query* with an *unsatisfiable query*.

Example 4.3 (satisfiability) Consider again the query Q_1 and the query

$$\mathsf{StarredIn}(x, y) \wedge \neg\mathsf{StarredIn}(x, y).$$

The latter is an *unsatisfiable query*, but it evaluates to the same probability as Q_1, that is, a *satisfiable query*; more concretely, both queries evaluate to the probability zero on the PDB $\mathcal{P}_c$. ◊

In a nutshell, the CWA forces a very flat representation, and as a consequence, it becomes impossible to even distinguish a satisfiable query from an unsatisfiable one by comparing their probabilities. Importantly, these counterintuitive results are not synthetic, i.e., they are observed in *real-world data* as well, as we will illustrate later.

4.1.2 An Unfounded Learning Loop

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. Probabilistic data (and knowledge) bases can be cast into this principled framework as follows.

Suppose we are building a probabilistic knowledge base from scratch. The first step of Bayesian learning is to come up with a prior belief about the facts in the database. Next, as we read the web, we incorporate more evidence into our distribution. For example, suppose we observe two web pages, d_a and d_b, and are interested in querying for Q_2 as defined above. Then, we may have

$$\mathrm{P}(Q_2) = 0.01, \qquad \mathrm{P}(Q_2 \mid d_a) = 0.09, \qquad \mathrm{P}(Q_2 \mid d_a, d_b) = 0.08,$$

that is, our prior belief is that the probability of Q_2 is 1%, but after observing the information on web page d_a, that probability becomes 9%. When additionally observing web page d_b, giving evidence to the contrary, the belief drops to 8%.

This sequence is a typical run of Bayesian learning. Unfortunately, it is not the mode of operation for large-scale PDBs as they currently function. A typical run would instead be

$$\mathrm{P}(Q_2) = 0, \qquad \mathrm{P}(Q_2 \mid d_a) = 0.09, \qquad \mathrm{P}(Q_2 \mid d_a, d_b) = 0.08,$$

The difference is subtle, but important. The first induction, from a belief of 0% to 9% is impossible to obtain from a single probability distribution P and violates the axioms of belief update. When Q_2 is impossible according to P, it remains impossible after observing evidence. Consequently, the Bayesian learning paradigm fails in practice. More precisely, given a PDB at time t, such systems gather data D^t to obtain a new model $\mathrm{P}^{t+1}(.) = \mathrm{P}^t(. \mid D^t)$. Systems continuously add facts f, that is, set $\mathrm{P}^{t+1}(f) > 0$, whereas previously $\mathrm{P}^t(f) = 0$; an impossible induction for Bayesian learning.

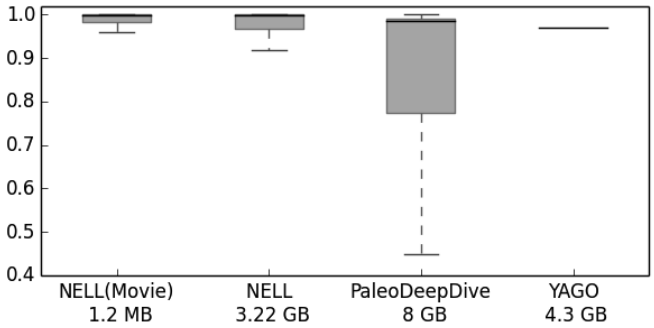


Figure 4.1: Box plot of probabilistic knowledge base probabilities with 2nd to 3rd quartile in gray. YAGO only provides an estimate of the mean probability per relation.

4.1.3 Knowledge Base Completion, Mining and Evaluation

The CWA permeates higher-level tasks that one is usually interested in performing on probabilistic databases. Consider for example the knowledge base completion task, where we want to learn new facts to add to the database, using the facts that are already present. A natural approach to knowledge base completion is to learn a probabilistic model from training data. Consider for example a probabilistic rule (De Raedt et al. 2015; Wang, Mazaitis, and Cohen 2013) of the form

$$\mathsf{StarCouples}(x, y) \xleftarrow{0.8} \mathsf{StarredIn}(x, z), \mathsf{StarredIn}(y, z), \mathsf{Couple}(x, y).$$

encoding the fact that if the query $\mathsf{StarredIn}(x, z), \mathsf{StarredIn}(y, z), \mathsf{Couple}(x, y)$ succeeds on a database, there is an 80% probability that we should derive the fact $\mathsf{StarCouples}(x, y)$. To evaluate the quality of this rule to predict the star couples relation, the standard approach would be to take the current probabilistic database together with labeled training data

$$\mathcal{D} = \{\mathsf{StarCouples}(\mathsf{willsmith}, \mathsf{jadasmith}), \mathsf{StarCouples}(\mathsf{pitt}, \mathsf{jolie})\}$$

and quantify the conditional likelihood of the rule (Sutton and McCallum 2011). Unfortunately, by the CWA, the rule predicts $\mathrm{P}(\mathsf{StarCouples}(\mathsf{willsmith}, \mathsf{jadasmith})) = 0$ because the fact $\mathsf{Couple}(\mathsf{willsmith}, \mathsf{jadasmith})$ is missing from the database. The rule gets the worst likelihood score of zero, regardless of its performance on other tuples in the training data. Indeed, our probabilistic database semantics tell us that the absence of a single tuple can make $\mathsf{StarCouples}(\mathsf{willsmith}, \mathsf{jadasmith})$ impossible, invalidating the entire rule, which is otherwise highly accurate.

Another high-level task is to mine frequent patterns in the knowledge base. Given the probabilistic database the goal would, for instance, be to find interesting patterns, such as the pattern that many couples star in the same movie, and report it to the data miner. Again, the CWA will underestimate the expected frequencies of these patterns, and stand in the way of progress (Galárraga, Teflioudi, Hose, and Suchanek 2013).

4.1.4 Truncating and Space Blow-up

Figure 4.1 shows the distribution of the probabilities in popular data and 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. 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). Moreover, note that the distribution of probabilities for such a closed-world database would be vastly different from the current ones, and instead be highly skewed towards zero. This issue of truncating and quadratic blow-up is inherent to probabilistic knowledge bases and has to be acknowledged in their semantics.

4.2 Open-World Probabilistic Databases

Our central observation is that large scale knowledge bases are *incomplete* by their nature and systems used to build such knowledge bases should incorporate these characteristics into the query semantics. A feasible approach in this context would be to relax the probabilities of facts that are not in the database to a default probability interval, which is very different from the closed-world assumption of PDBs, which requires the probabilities of such facts to be zero. Our proposal on open-world probabilistic databases builds on the theory of imprecise probabilities to allow default, interval-based probabilities for the atoms that are not in the database. Syntactically, an open-world probabilistic database is a pair of a probabilistic database and a predefined threshold value.

Definition 4.4 (syntax) An *open-world probabilistic database (OpenPDB)* is a pair $\mathcal{G} = (\mathcal{P}, \lambda)$, where $\mathcal{P}$ is a probabilistic database and $\lambda \in [0, 1]$. ◊

The semantics of OpenPDBs is based on completing probabilistic databases. Intuitively, an OpenPDB denotes a partial specification over a vocabulary and needs to be completed by assigning a probability value from an interval $[0, \lambda]$ to each of the open tuples.

Definition 4.5 (completion) A λ-*completion* of a probabilistic database $\mathcal{P}$ is another probabilistic database that is obtained as follows. For each atom t that does not appear in $\mathcal{P}$, we add an atom $\langle t : p \rangle$ to $\mathcal{P}$ for some $p \in [0, \lambda]$. ◊

An OpenPDB induces a set of PDBs, each of which differs in the probabilities of the open tuples. Therefore, while a closed probabilistic database induces a unique probability distribution, an OpenPDB induces a (credal) set of probability distributions. A *credal set* is a closed convex set of probability distributions over a shared set of random variables.

Definition 4.6 (OpenPDBs) An open probabilistic database $\mathcal{G} = (\mathcal{P}, \lambda)$ induces a credal set of probability distributions $\mathrm{K}_\mathcal{G}$ such that distribution P belongs to $\mathrm{K}_\mathcal{G}$ if and only if P is induced by some λ-completion of the PDB $\mathcal{P}$. ◊

Table 4.2: The OpenPDB $\mathcal{G}_c = (\mathcal{P}_c, 0.5)$ induces an infinite set of PDBs. Rows depicted in orange color represent open tuples that can take on any rational probability value from the default probability interval $[0, 0.5]$.

StarredIn		P
willsmith	ali	0.9
willsmith	sharktale	0.8
jadasmith	ali	0.6
arquette	scream	0.7
pitt	mrmssmith	0.5
jolie	mrmssmith	0.7
jolie	sharktale	0.9
pitt	ali	[0, 0.5]
pitt	fightclub	[0, 0.5]
arquette	fightclub	[0, 0.5]
...	...	[0, 0.5]

Couple		P
arquette	cox	0.6
pitt	jolie	0.8
thornton	jolie	0.6
pitt	aniston	0.9
kunis	kutcher	0.7
willsmith	jadasmith	[0, 0.5]
arquette	jolie	[0, 0.5]
pitt	kutcher	[0, 0.5]
...	...	[0, 0.5]

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.

Example 4.7 Consider the PDB $\mathcal{P}_c$ given before. Then, the pair $\mathcal{G}_c = (\mathcal{P}_c, 0.5)$ denotes an OpenPDB where open tuples can have the probability *at most* 0.5. Clearly, there are infinitely many possible completions of $\mathcal{G}_c$. Consider, for instance, the following completions

$$\mathcal{P}_0 = \mathcal{P}_c \cup \{\langle t : 0\rangle \mid t \textit{ is an open atom in the PDB } \mathcal{P}_c\},$$
$$\mathcal{P}_{0.5} = \mathcal{P}_c \cup \{\langle t : 0.5\rangle \mid t \textit{ is an open atom in the PDB } \mathcal{P}_c\}.$$

These completions are special since they uniformly set all of open atoms to the *same probability* value. Observe, furthermore, that these completions induce different probability distributions, both of which belong to $\mathrm{K}_{\mathcal{G}_c}$. ◊

Query semantics now has to take into account sets of probability distributions, and provide query probabilities in terms of *upper and lower probability values*.

Definition 4.8 (query semantics) Let Q be a Boolean query and $\mathcal{G}$ be an OpenPDB. The *probability interval of* Q in the OpenPDB $\mathcal{G}$ is defined as $\mathrm{K}_{\mathcal{G}}(Q) = [\underline{\mathrm{P}}_{\mathcal{G}}(Q), \overline{\mathrm{P}}_{\mathcal{G}}(Q)]$, where

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

We will discuss the consequences of this semantics in more detail and compare it with the semantics of PDBs, in the next section. At this point, we confine ourselves to highlighting the basic semantic properties of OpenPDBs.

Example 4.9 Consider the OpenPDB $\mathcal{G}_c = (\mathcal{P}_c, 0.5)$ and suppose that an atom t is open in OpenPDB $\mathcal{G}_c$. We then have $\underline{\mathrm{P}}_{\mathcal{G}_c}(t) = 0$ and $\overline{\mathrm{P}}_{\mathcal{G}_c}(t) = \lambda$ by the query semantics. For instance, the ground query $Q = \mathsf{StarredIn}(\mathsf{pitt}, \mathsf{fightclub})$ evaluates to the probability

0 in PDBs, as the probability of the tuple $\mathsf{StarredIn}(\mathsf{pitt}, \mathsf{fightclub})$ is set to 0 by the CWA. As for $\mathcal{G}_c$, it is easy to see that

$$\underline{\mathrm{P}}_{\mathcal{G}_c}(Q) = 0 \quad \text{and} \quad \overline{\mathrm{P}}_{\mathcal{G}_c}(Q) = 0.5.$$

The lower probability of Q remains the same due to the completion that assigns all open atoms the probability 0, while the upper probability increases due the following completion $\mathcal{P}_{0.5}$ ◊

The open-world assumption has been introduced as the opposite of the CWA (Reiter 1978). 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, 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 (J. Y. Halpern 2003), credal networks (Cozman 2000). Moreover, to assume a default probability interval is clearly a form of default reasoning (Reiter 1980).

4.2.1 Open-World Probabilistic Databases in Practice

We discuss the implications of the open-world setting before moving to a detailed technical analysis. First, we go through the motivating examples provided in Section 4.1, and highlight the differences in OpenPDBs.

Example 4.10 (specificity) Recall that both of the following queries

$$Q_1(\mathsf{pitt}, \mathsf{jolie}) = \mathsf{StarredIn}(\mathsf{pitt}, z), \mathsf{StarredIn}(\mathsf{jolie}, z), \mathsf{Couple}(\mathsf{pitt}, \mathsf{jolie}),$$
$$Q_2 = \mathsf{StarredIn}(x, z), \mathsf{StarredIn}(y, z), \mathsf{Couple}(x, y),$$

evaluate to the same probability in the PDB $\mathcal{P}_c$ as explained in Example 4.1. We already noted that $Q_1(\mathsf{pitt}, \mathsf{jolie})$ entails Q_2, leading us to expect that $\mathrm{P}(Q_2) > \mathrm{P}(Q_1(\mathsf{pitt}, \mathsf{jolie}))$ assuming our knowledge is not complete. This is indeed the case for the OpenPDB $\mathcal{G}_c = (\mathcal{P}_c, 0.5)$ for upper probabilities. It holds that $\overline{\mathrm{P}}_{\mathcal{G}_c}(Q_2) > \overline{\mathrm{P}}_{\mathcal{G}_c}(Q_1(\mathsf{pitt}, \mathsf{jolie}))$ since there are many worlds with non-zero probability that entail Q_2 but not $Q_1(\mathsf{pitt}, \mathsf{jolie})$. Notice that the lower probabilities remain unchanged. ◊

This shows that, in the open-world, it becomes possible to distinguish a *query* from a particular *instance of this query*, by comparing their respective upper probabilities.

Example 4.11 (support) Consider now again the queries $Q_1(\mathsf{willsmith}, \mathsf{jadasmith})$ and $Q_1(\mathsf{thornton}, \mathsf{aniston})$. Despite the fact that the first query has more support than the second one in $\mathcal{P}_c$, both queries evaluate to the probability 0 in $\mathcal{P}_c$, as identified in Example 4.2. In the OpenPDB $\mathcal{G}_c = (\mathcal{P}_c, 0.5)$, we obtain

$$\overline{\mathrm{P}}(Q_1(\mathsf{willsmith}, \mathsf{jadasmith})) > \overline{\mathrm{P}}(Q_1(\mathsf{thornton}, \mathsf{aniston})) > 0,$$

meaning that we can distinguish the queries in terms of the relative support in the data.◊

Clearly, in the open-world setting, it becomes even possible to distinguish queries that do not have a matching answer in the database since such queries typically have different levels of support in the database. We have also observed that an unsatisfiable query is in some cases as likely as a satisfiable one in the closed-world. How are such queries evaluated in the open-world?

Example 4.12 (satisfiability) Consider again the query Q_1 and the unsatisfiable query $\mathsf{StarredIn}(x, y) \wedge \neg\mathsf{StarredIn}(x, y)$. Recall that both queries evaluate to the probability zero on the PDB $\mathcal{P}_c$. In the open-world setting, the upper probability of a satisfiable query will always be greater than the upper probability of an unsatisfiable query unless all the matches for the query are explicitly set to the probability 0 by the PDB. Moreover, any unsatisfiable query will still have a zero upper probability. ◊

These synthetic examples underline the difference in the semantics of PDBs and OpenPDBs. However, do we really encounter similar examples in the real-world data, which can benefit from an open-world perspective? To elaborate on this question, we have extracted a portion from NELL concerning movies, actors, directors, etc. We conclude this section with this real-world example.

Example 4.13 (real-world data) Consider the following queries constructed based on a portion of the NELL database:

$Q_1 \coloneqq \mathsf{Actor}(\mathsf{pattinson}) \wedge \mathsf{Workedfor}(\mathsf{pattinson}, \mathsf{hardwicke}) \wedge \mathsf{Director}(\mathsf{hardwicke}).$

$Q_2 \coloneqq \exists x \mathsf{Actor}(x) \wedge \mathsf{StarredIn}(x, \mathsf{trainspotting}) \wedge \mathsf{Movie}(\mathsf{trainspotting}) \wedge \neg\mathsf{Director}(x).$

$Q_3 \coloneqq \exists x \mathsf{Actor}(\mathsf{pattinson}) \wedge \mathsf{Workedfor}(\mathsf{pattinson}, x) \wedge \mathsf{Director}(x).$

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_1 entails Q_3, and posing these queries in the open-world setting, we indeed obtain $\overline{\mathrm{P}}(Q_3) > \overline{\mathrm{P}}(Q_1)$ for any non-zero threshold λ. For instance, $\overline{\mathrm{P}}(Q_3) = 0.82$ and $\overline{\mathrm{P}}(Q_1) = 0.51$ for $\lambda = 0.3$. The query Q_2 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_2 in OpenPDBs yields $\overline{\mathrm{P}}(Q_2) = 0.98$ and $\overline{\mathrm{P}}(Q_2) = 0.78$ with thresholds 0.7 and 0.3, respectively. These answers are clearly more in line with what one would expect. ◊

4.2.2 Query Answering in Open-World Probabilistic Databases

We study the following decision problems that extend probabilistic query evaluation to consider lower and upper probabilities for queries.

Definition 4.14 (upper, lower probabilistic query evaluation) Given an OpenPDB $\mathcal{G}$, a query Q and a value $p \in [0, 1)$, *upper probabilistic query evaluation*, denoted $\overline{\mathsf{PQE}}$, is to decide whether $\overline{\mathrm{P}}_{\mathcal{G}}(Q) > p$ and *lower probabilistic query evaluation* denoted $\underline{\mathsf{PQE}}$, is to decide whether $\underline{\mathrm{P}}_{\mathcal{G}}(Q) > q$. Both $\overline{\mathsf{PQE}}$ and $\underline{\mathsf{PQE}}$ are parametrized with a particular query language; thus, we write $\overline{\mathsf{PQE}}(\mathsf{Q})$ (respectively, $\underline{\mathsf{PQE}}(\mathsf{Q})$) to define $\overline{\mathsf{PQE}}$ (respectively, $\underline{\mathsf{PQE}}$) on the class of Q queries. ◊

OpenPDBs model an infinite set of PDBs, and it may seem like an unsurmountable task to efficiently compute the probability intervals $\mathrm{K}_{\mathcal{G}}(Q)$. As we will prove later, the problem can be simplified to consider only *extremal* probability distributions that are obtained by setting the probability values of all elementary events to one of the extreme points.

Definition 4.15 Let $\mathcal{G} = (\mathcal{P}, \lambda)$ be an arbitrary OpenPDB; we call a probability distribution $\mathrm{P} \in \mathrm{K}_{\mathcal{G}}$ an *extremal distribution* if for all open atoms t, either $\mathrm{P}(t) = \lambda$ or $\mathrm{P}(t) = 0$ holds. ◊

By Theorem 4.16, to compute the upper and lower probability bounds, it is sufficient to consider the distributions, where open atoms can take on the probability λ or 0 , i.e., no intermediate choices need to be examined.

Theorem 4.16 *Let $\mathcal{G}$ be an arbitrary OpenPDB and Q an* FO *query. There exist extremal distributions $\underline{\mathrm{P}}, \overline{\mathrm{P}} \in \mathrm{K}_{\mathcal{G}}$ such that $\underline{\mathrm{P}}(Q) = \underline{\mathrm{P}}_{\mathcal{G}}(Q)$, and $\overline{\mathrm{P}}(Q) = \overline{\mathrm{P}}_{\mathcal{G}}(Q)$.*

Proof. First, note that the functions $\overline{\mathrm{P}}_{\mathcal{G}} : \mathsf{FO} \mapsto [0, \lambda]$ and $\underline{\mathrm{P}}_{\mathcal{G}} : \mathsf{FO} \mapsto [0, \lambda]$ are *well-defined* w.r.t. the set $\mathrm{K}_{\mathcal{G}}$, i.e., the existence of a *maximum* (resp., *minimum*) is ensured by the properties of credal sets. We need to show that the maximal (resp., minimal) probabilities of queries can always be obtained from the extreme probability distributions. We prove the claim only for $\overline{\mathrm{P}}$ as $\underline{\mathrm{P}}$ can be treated analogously.

To simplify the proof, we use the lineage representation of the database atoms, which can be realized simply by introducing a propositional literal p_t for every atom t. Similarly, we focus on the lineage of the query, which can be obtained by first grounding the query and then converting it into a propositional formula by replacing every tuple in the ground query with its lineage. It is well-known that every first-order query relative to a finite structure has a corresponding propositional lineage representation and thus our assumption is without loss of generality. Since any propositional formula is equivalent to a formula in 3CNF, we can further assume that the lineage is in 3CNF. Moreover, for simplicity we assume that the CNF contains *exactly* three clauses. Thus, it suffices to prove the claim for 3CNF formulas $\varphi = c_1 \wedge \ldots \wedge c_n$ where $c_i = {}^{(\neg)}l_{i_1} \vee {}^{(\neg)}l_{i_2} \vee {}^{(\neg)}l_{i_3}$.

Suppose that there is a probability distribution P, where the probabilities of k (positive) literals in φ are set to intermediate probability values from the interval $(0, \lambda)$. We prove that, for each such literal l, there is (at least) one extreme assignment to l that does not decrease the probability of φ. Formally, given P, we define two new probability distributions $\mathrm{P}^{l=\lambda}$ and $\mathrm{P}^{l=0}$ such that $\mathrm{P}^{l=\lambda}(l_{ij}) = \mathrm{P}(l_{ij})$ and $\mathrm{P}^{l=0}(l_{ij}) = \mathrm{P}(l_{ij})$ for all l_{ij} different from l and $\mathrm{P}^{l=\lambda}(l) = \lambda$, and $\mathrm{P}^{l=0}(l) = 0$.

Claim. Either $\mathrm{P}^{l=\lambda}(\varphi) \geq \mathrm{P}(\varphi)$, or $\mathrm{P}^{l=0}(\varphi) \geq \mathrm{P}(\varphi)$ holds.

To prove the claim, suppose that $\mathrm{P}^{l=\lambda}(\varphi) < \mathrm{P}(\varphi)$, i.e., the probability of $\varphi = c_1 \wedge \ldots \wedge c_n$ decreases if we increase the probability of l to λ. We make a case analysis.

Case 1. Assume that the literal l appears only positively in φ. This immediately leads to a contradiction since, if for every clause c_i, l appears positively, then the probability of φ is clearly *monotone* in l. Thus, $\mathrm{P}^{l=\lambda}(\varphi) \geq \mathrm{P}(\varphi)$.

Case 2. Assume that the literal l appears only negatively in φ. If for every clause c_i, l appears negatively, then the probability of φ is *antitone* in l. This immediately

implies that $\mathrm{P}^{l=0}(\varphi) \geq \mathrm{P}(\varphi)$ since further decreasing the probability of l increases the probability of φ.

Case 3. Assume that the literal l appears both positively and negatively in φ. We can summarize all clauses where l appears positively with the formula

$$l \ \vee \ \Big(\underbrace{\big((\neg)u_1 \vee (\neg)u_2\big) \ \wedge \ \ldots \ \wedge \ \big((\neg)u_r \vee (\neg)u_{r+1}\big)}_{\Delta_1} \Big),$$

and similarly the clauses where l appears negatively with the formula

$$\neg l \ \vee \ \Big(\underbrace{\big((\neg)v_1 \vee (\neg)v_2\big) \ \wedge \ \ldots \ \wedge \ \big((\neg)v_s \vee (\neg)v_{s+1}\big)}_{\Delta_2} \Big).$$

Moreover, let Δ_3 be the conjunction of all clauses where the literal l does not appear. Thus, we obtain $(l \vee \Delta_1) \wedge (\neg l \vee \Delta_2) \wedge \Delta_3$ as a rewriting of φ. We can compute the probability of $\varphi = (l \vee \Delta_1) \wedge (\neg l \vee \Delta_2) \wedge \Delta_3$ by applying *inclusion-exclusion* (see the description of Step 5 in Algorithm 1) as

$$\begin{aligned}\mathrm{P}(\varphi) = &- \mathrm{P}(l \vee \Delta_1) - \mathrm{P}(\neg l \vee \Delta_2) - \mathrm{P}(\Delta_3) \\ &+ \mathrm{P}((l \vee \Delta_1) \vee (\neg l \vee \Delta_2)) + \mathrm{P}((l \vee \Delta_1) \vee \Delta_3) + \mathrm{P}((\neg l \vee \Delta_2) \vee \Delta_3) \\ &- \mathrm{P}((l \vee \Delta_1) \vee (\neg l \vee \Delta_2) \vee \Delta_3).\end{aligned}$$

Note that $\mathrm{P}((l \vee \Delta_1) \vee (\neg l \vee \Delta_2)) = \mathrm{P}((l \vee \Delta_1) \vee (\neg l \vee \Delta_2) \vee \Delta_3) = 1$, which yields

$$\mathrm{P}(\varphi) = - \mathrm{P}(l \vee \Delta_1) - \mathrm{P}(\neg l \vee \Delta_2) - \mathrm{P}(\Delta_3) + \mathrm{P}(l \vee \Delta_1 \vee \Delta_3) + \mathrm{P}(\neg l \vee \Delta_2 \vee \Delta_3).$$

We can now decompose the terms that are independent, for instance, l and Δ_1 are independent from each other; thus, we can decompose the disjunction $(l \vee \Delta_1)$ using the independence assumption as $\mathrm{P}(l \vee \Delta_1) = (1 - (1 - \mathrm{P}(l)) \cdot (1 - \mathrm{P}(\Delta_1)))$ and apply other well-known simplifications to obtain

$$\begin{aligned}\mathrm{P}(\varphi) = &- \mathrm{P}(l \vee \Delta_1) - \mathrm{P}(\neg l \vee \Delta_2) - \mathrm{P}(\Delta_3) + \mathrm{P}(l \vee \Delta_1 \vee \Delta_3) + \mathrm{P}(\neg l \vee \Delta_2 \vee \Delta_3) \\ = &- \Big(1 - (1 - \mathrm{P}(l)) \cdot (1 - \mathrm{P}(\Delta_1))\Big) - \Big(1 - (1 - \mathrm{P}(\neg l)) \cdot (1 - \mathrm{P}(\Delta_2))\Big) - \mathrm{P}(\Delta_3) \\ &+ \Big(1 - (1 - \mathrm{P}(l)) \cdot (1 - \mathrm{P}(\Delta_1 \vee \Delta_3))\Big) + \Big(1 - (1 - \mathrm{P}(\neg l)) \cdot (1 - \mathrm{P}(\Delta_2 \vee \Delta_3))\Big) \\ = &- \Big(1 - (1 - \mathrm{P}(l)) \cdot (1 - \mathrm{P}(\Delta_1))\Big) - \Big(1 - \mathrm{P}(l) \cdot (1 - \mathrm{P}(\Delta_2))\Big) - \Big(\mathrm{P}(\Delta_3)\Big) \\ &+ \Big(1 - (1 - \mathrm{P}(l)) \cdot (1 - \mathrm{P}(\Delta_1 \vee \Delta_3))\Big) + \Big(1 - \mathrm{P}(l) \cdot (1 - \mathrm{P}(\Delta_2 \vee \Delta_3))\Big).\end{aligned}$$

We simplify $\mathrm{P}(\varphi)$ further and obtain

$$\begin{aligned}
\mathrm{P}(\varphi) = & -\Big(1-(1-\mathrm{P}(l))\cdot(1-\mathrm{P}(\Delta_1))\Big)-\Big(1-\mathrm{P}(l)\cdot(1-\mathrm{P}(\Delta_2))\Big)-\Big(\mathrm{P}(\Delta_3)\Big)\\
& +\Big(1-(1-\mathrm{P}(l))\cdot(1-(1-(1-\mathrm{P}(\Delta_1))\cdot(1-\mathrm{P}(\Delta_3))))\Big)\\
& +\Big(1-\mathrm{P}(l)\cdot(1-(1-(1-\mathrm{P}(\Delta_2))\cdot(1-\mathrm{P}(\Delta_3))))\Big)\\
= & -\Big(\mathrm{P}(\Delta_1)+\mathrm{P}(l)-\mathrm{P}(l)\,\mathrm{P}(\Delta_1)\Big)-\Big(1-\mathrm{P}(l)+\mathrm{P}(l)\,\mathrm{P}(\Delta_2)\Big)-\Big(\mathrm{P}(\Delta_3)\Big)\\
& +\Big(\mathrm{P}(\Delta_3)+\mathrm{P}(\Delta_1)-\mathrm{P}(\Delta_1)\cdot\mathrm{P}(\Delta_3)+\mathrm{P}(l)-\mathrm{P}(l)\cdot\mathrm{P}(\Delta_3)\\
& \qquad -\mathrm{P}(l)\cdot\mathrm{P}(\Delta_1)-\mathrm{P}(l)\cdot\mathrm{P}(\Delta_1)\cdot\mathrm{P}(\Delta_3)\Big)\\
& +\Big(1-\mathrm{P}(l)+\mathrm{P}(l)\cdot\mathrm{P}(\Delta_3)+\mathrm{P}(l)\cdot\mathrm{P}(\Delta_2)+\mathrm{P}(l)\cdot\mathrm{P}(\Delta_2)\cdot\mathrm{P}(\Delta_3)\Big).
\end{aligned}$$

After applying cancellations, we obtain $\mathrm{P}(\varphi)=\mathrm{P}(\Delta_3)\cdot\big(\mathrm{P}(l)\cdot(\mathrm{P}(\Delta_1)-\mathrm{P}(\Delta_2))-\mathrm{P}(\Delta_1)\big)$. Consider now the probability distributions $\mathrm{P}^{l=0}$ and $\mathrm{P}^{l=\lambda}$. It is easy to see that

$$\begin{aligned}
\mathrm{P}(\Delta_1)=\mathrm{P}^{l=\lambda}(\Delta_1)=\mathrm{P}^{l=0}(\Delta_1)=\alpha,\\
\mathrm{P}(\Delta_2)=\mathrm{P}^{l=\lambda}(\Delta_2)=\mathrm{P}^{l=0}(\Delta_2)=\beta,\\
\mathrm{P}(\Delta_3)=\mathrm{P}^{l=\lambda}(\Delta_3)=\mathrm{P}^{l=0}(\Delta_3)=\gamma
\end{aligned}$$

as i) both $\mathrm{P}^{l=0}$, and $\mathrm{P}^{l=\lambda}$ are equivalent to the given probability distribution P *modulo* the probability assignments for l and ii) Δ_1, Δ_2, Δ_3 are all probabilistically independent from l. Recall that $\mathrm{P}^{l=\lambda}(\varphi)<\mathrm{P}(\varphi)$, and thus

$$\gamma\cdot\big(\mathrm{P}^{l=\lambda}(l)\cdot(\alpha-\beta)-\alpha\big)<\gamma\cdot\big(\mathrm{P}(l)\cdot(\alpha-\beta)-\alpha\big),$$

which holds if and only if either $\mathrm{P}^{l=\lambda}(l)>\mathrm{P}(l)$, or $\alpha<\beta$. It is easy to see that the former would immediately contradict with our assumption. Suppose that $\alpha<\beta$ holds and consider the distribution $\mathrm{P}^{l=0}$. As before, we can compute $\mathrm{P}^{l=0}(\varphi)$ as

$$\begin{aligned}
\mathrm{P}^{l=0}(\varphi)&=\gamma\cdot\big(\mathrm{P}^{l=\lambda}(l)\cdot(\alpha-\beta)-\alpha\big)=\gamma\cdot\big(0\cdot(\alpha-\beta)-\alpha\big)=-\gamma\cdot\alpha\\
&=\gamma\cdot\big(\mathrm{P}(l)\cdot(\alpha-\alpha)-\alpha\big)\geq\gamma\cdot\big(\mathrm{P}(l)\cdot(\alpha-\beta)-\alpha\big)=\mathrm{P}(\varphi).
\end{aligned}$$

This shows that if $\mathrm{P}^{l=\lambda}(\varphi)<\mathrm{P}(\varphi)$ then $\mathrm{P}^{l=0}(\varphi)\geq\mathrm{P}(\varphi)$. Therefore, this concludes our case analysis and proves the claim.

Observe that this argument can be applied repeatedly until there is no such literal left, i.e., all literals are assigned a probability value that is extreme. Clearly, this procedure terminates, and implies that, for any probability distribution that is maximal, but not extreme, we can find a corresponding extreme distribution that is also maximal. □

As a consequence of Theorem 4.16 query answering in OpenPDBs can be performed by focusing on extremal distributions. More precisely, there are exponentially many extremal distributions, each of which sets the probability of (at least) one atom to a different extreme.

As for PDBs, probabilistic query evaluation in OpenPDBs is a computationally demanding task. We show, however, that probabilistic query evaluation in OpenPDBs does not incur a computational overhead compared to probabilistic query evaluation in PDBs if we restrict ourselves to unions of conjunctive queries. In what follows, we discuss the implications of these properties on these queries and present two different approaches for computing the query probabilities.

A Naïve Reduction to PDBs for Unions of Conjunctive Queries

Theorem 4.16 suggests a naïve query answering algorithm: generate all extreme distributions P, compute $\mathrm{P}(Q)$, and report the minimum and maximum. Obviously, this procedure is ensured to terminate. Nevertheless, this naïve approach is exponential in the number of open-world atoms.

For unions of conjunctive queries, the monotone satisfaction relation allows us to further simplify query evaluation. In essence, we can simply choose the minimal (resp. maximal) bound for every atom and the resulting probability for the UCQ is ensured to be minimal (resp. maximal). Thus, the lower probabilities of conjunctive queries in OpenPDBs can be computed using a standard PDB algorithm. 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 4.17 *Let $\mathcal{G} = (\mathcal{P}, \lambda)$ be an arbitrary OpenPDB, Q a UCQ and $\mathcal{P}_\lambda$ the completion that sets the probabilities of all open tuples to λ. Then, it holds that*

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

Proof. We prove the result for the upper bound: $\overline{\mathrm{P}}_\mathcal{G}(Q) = \mathrm{P}_{\mathcal{P}_\lambda}(Q)$. The proof for the lower bound $\underline{\mathrm{P}}_\mathcal{G}(Q)$, can be obtained analogously. It is easy to see that the function $\overline{\mathrm{P}}_\mathcal{G} : \mathsf{UCQ} \mapsto [0, \lambda]$ is monotone. By Definition 4.6, we know that $\mathrm{P}_{\mathcal{P}_\lambda} \in \mathrm{K}_\mathcal{G}$. Thus, we obtain $\overline{\mathrm{P}}_\mathcal{G}(Q) \geq \mathrm{P}_{\mathcal{P}_\lambda}(Q)$. To show the other direction, i.e., $\overline{\mathrm{P}}_\mathcal{G}(Q) \leq \mathrm{P}_{\mathcal{P}_\lambda}(Q)$, assume by contradiction that $\overline{\mathrm{P}}_\mathcal{G}(Q) > \mathrm{P}_{\mathcal{P}_\lambda}(Q)$. Then, by Theorem 4.16, there exists a PDB $\hat{\mathcal{P}}$ that uses only the extreme points for the open tuples (i.e., induces an extreme distribution) and that satisfies $\mathrm{P}_{\hat{\mathcal{P}}}(Q) = \overline{\mathrm{P}}_\mathcal{G}(Q) > \mathrm{P}_{\mathcal{P}_\lambda}(Q)$. Since $\mathcal{P}_\lambda$ and $\hat{\mathcal{P}}$ induce different distributions, there must exist at least one atom t for which $\mathrm{P}_{\hat{\mathcal{P}}}(t) = 0$ and $\mathrm{P}_{\mathcal{P}_\lambda}(t) = \lambda$. Then, by the monotonicity of $\overline{\mathrm{P}}$ on unions of conjunctive queries, it follows that $\mathrm{P}_{\hat{\mathcal{P}}}(Q) \leq \mathrm{P}_{\mathcal{P}_\lambda}(Q)$, which leads to a contradiction. □

Notice that the construction in Theorem 4.17 is efficient. It adds all the atoms to 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 4.1.4, 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.

The Quest for Efficient Probabilistic Query Evaluation

To achieve a practical query evaluation algorithm for unions of conjunctive queries, we are interested in finding methods to compute the upper probability of a UCQ without

the need of explicit grounding. Observe first that in OpenPDBs, we can easily recover the upper (resp. lower) probability of a query from the lower (resp.upper) probability of its negation as shown next.

Lemma 4.18 *Let $\mathcal{G} = (\mathcal{P}, \lambda)$ be an OpenPDB and Q a first-order query; it holds that $\overline{\mathrm{P}}_{\mathcal{G}}(Q) = 1 - \underline{\mathrm{P}}_{\mathcal{G}}(\neg Q)$ and $\underline{\mathrm{P}}_{\mathcal{G}}(Q) = 1 - \overline{\mathrm{P}}_{\mathcal{G}}(\neg Q)$.*

Proof. This is a simple consequence of the query semantics, which asserts that either $\mathcal{D} \models Q$ or $\mathcal{D} \models \neg Q$ must hold for any database $\mathcal{D}$ and query Q. On this level, the semantics forces completeness, and therefore, it is never the case that neither $\mathcal{D} \models Q$ nor $\mathcal{D} \models \neg Q$ holds. By this argument, for any probability distribution P, it holds that $\mathrm{P}(Q) = 1 - \mathrm{P}(\neg Q)$ and thus any probability distribution is closed under complement relative to the queries. Using this and the existence of maximal and minimal distributions in OpenPDBs, we obtain

$$\begin{aligned}
\overline{\mathrm{P}}_{\mathcal{G}}(Q) &= \max\{\mathrm{P}(Q) \mid \mathrm{P} \in \mathrm{K}_{\mathcal{G}}\} \\
1 - \overline{\mathrm{P}}_{\mathcal{G}}(Q) &= 1 - \max\{\mathrm{P}(Q) \mid \mathrm{P} \in \mathrm{K}_{\mathcal{G}}\} \\
&= \min\{1 - \mathrm{P}(Q) \mid \mathrm{P} \in \mathrm{K}_{\mathcal{G}}\} \\
&= \min\{\mathrm{P}(\neg Q) \mid \mathrm{P} \in \mathrm{K}_{\mathcal{G}}\} \\
&= \underline{\mathrm{P}}_{\mathcal{G}}(\neg Q).
\end{aligned}$$

Given the OpenPDB $\mathcal{G}$ and the query Q, let $\overline{\mathrm{P}}$ (respectively, $\underline{\mathrm{P}}$) be the extremal distribution from $\mathrm{K}_{\mathcal{G}}$ that maximizes (respectively, minimizes) the query probability; that is, $\overline{\mathrm{P}}_{\mathcal{G}}(Q) = \overline{\mathrm{P}}(Q)$ and $\underline{\mathrm{P}}_{\mathcal{G}}(Q) = \underline{\mathrm{P}}(Q)$ where P is an extremal distribution in $\mathrm{K}_{\mathcal{G}}$ that maximizes the query probability. □

We now present an algorithm; called $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$: it performs operations on monotone ∀CNF formulas, which are unions of CNF formulas, as we describe later. Given a UCQ Q, we consider its negation $\neg Q$, which is equivalent to a query in monotone ∀CNF, and Lemma 4.18 states the immediate connection.

Algorithm 1 computes upper bounds of CNF queries on OpenPDBs (as lower bounds can be computed with any closed-world PDB algorithm). CNF formulas are dual to unions of conjunctive queries. Hence, this algorithm also computes the probability of UCQ queries by Lemma 4.18: practically, this is achieved by taking the negation of the query and all literals, and returning the complement of the resulting probability.

Preprocessing. The algorithm to be presented 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 can be done in polynomial time in data complexity and thus it is efficient.

Preprocessing is needed for several reasons; most importantly, in order to capture all safe queries by an algorithm. We will explain how this preprocessing helps after the details of the algorithm are presented. We first give the details of the preprocessing.

Definition 4.19 (shattering, ranking) A first-order query Q is *shattered* if it does not contain any constants. A first-order query Q is *ranked* if there exists a total order $\prec$ on its variables such that for every atom $\mathsf{R}(\vec{x})$ of arity $k \geq 2$, whenever x_i, x_j occur

in $\mathsf{R}(\vec{x})$ and x_i occurs before x_j then $x_i \prec x_j$; in particular, no atom contains the same variable twice. ◊

We do not give a detailed overview of the preprocessing that consists of shattering and ranking. Instead, we briefly explain the process of ranking on a simple example.

Example 4.20 Consider the query $\exists x, y\ \mathsf{S}(x, y) \wedge \mathsf{S}(y, x)$, which is not ranked, since the variables x and y occur in different orders in the same predicate. To rank this query, we first split the predicate S into three predicates $\mathsf{S}_{x\prec y}$, $\mathsf{S}_{y\prec x}$ and $\mathsf{S}_{x=x}$. We then define a total order ϱ on the database constants (say a and b) and split the S-atoms in the PDB such that all occurrences of

- $\mathsf{S}(a, b)$ is replaced with $\mathsf{S}_{x\prec y}(a, b)$ if $a \prec b$,
- $\mathsf{S}(a, b)$ is replaced with $\mathsf{S}_{y\prec x}(b, a)$ if $b \prec a$,
- $\mathsf{S}(a, a)$ is replaced with $\mathsf{S}_{x=x}(a)$,
- $\mathsf{S}(b, b)$ is replaced with $\mathsf{S}_{x=x}(b)$.

This ensures that all appearances of the variables in some atom respects the order. Then, the ranking of the query $\exists x, y\ \mathsf{S}(x, y) \wedge \mathsf{S}(y, x)$ is given as

$$\exists x, y\ (\mathsf{S}_{x\prec y}(x, y) \wedge \mathsf{S}_{y\prec x}(x, y)) \vee \exists x \mathsf{S}_{x=x}(x)$$

Intuitively, this preprocessing partitions the predicates and the corresponding atoms in the database with respect to some ordering. It is easy to see that this transformation preserves the semantics; for details, see (Dalvi and Suciu 2012). ◊

It has been shown that the preprocessing does not affect the probability computation in PDBs: let Q be a query, $\mathcal{P}$ be a PDB, and Q^r, $\mathcal{P}^r$, their rankings. Then, it holds that $\mathrm{P}_{\mathcal{P}}(Q) = \mathrm{P}_{\mathcal{P}^r}(Q^r)$. This clearly translates to OpenPDBs since once a λ-completion is chosen for all open atoms, we obtain a single ranked PDB.

Ranking can be done in linear time in PDBs, but for OpenPDBs, this is unfortunately not the case, since we also have to consider the open atoms. Thus, in the worst case, the polynomial blow-up seems to be *unavoidable* in OpenPDBs. However, ranking is only needed for repeating relation symbols, i.e., if the query is self-join free, then this is not needed. Therefore, this preprocessing can be limited to repeating relation symbols so as to avoid to polynomial blow-up as much as possible. In the presented algorithm, we assume that the query and the PDB are preprocessed in this way.

Lifted Inference Algorithm. Algorithm 1 is an adaptation of the $\mathsf{Lift}^{\mathsf{R}}$ algorithm presented in (Gribkoff, Van den Broeck, and Suciu 2014b), which goes back to the algorithm of (Dalvi and Suciu 2012). This algorithm is called $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$, where O stands for open.

Step 0. Recall that the given UCQ is negated in the preprocessing to obtain a ∀CNF query Q. As a result, all atoms appear negatively in Q. The *base case* of the algorithm applies when the query is simply a negated ground atom $\neg t$. In this case the probability of the query is trivial to compute: if the atom appears in the PDB with a probability p, then the algorithm returns $(1 - p)$; otherwise, it is an open atom and the algorithm returns $(1 - \lambda)$.

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

Require: A negated UCQ Q, PDB $\mathcal{P}$, threshold value λ, and domain Δ.
Ensure: The lower probability $\underline{\mathrm{P}}(Q)$ in the OpenPDB $(\mathcal{P}, \lambda)$ over domain Δ.

Step 0 *Base of Recursion*
  if $Q = \neg t$, where t is a ground atom **then**
    if $\langle t : p \rangle \in \mathcal{P}$ **then return** $(1 - p)$
    else return $(1 - \lambda)$ ▷ Default values for open atoms
Step 1 *Rewriting of Query*
  Convert Q to UCNF: $Q_{\mathsf{UCNF}} \coloneqq (\forall \vec{x}\, Q_1) \vee \ldots \vee (\forall \vec{y}\, Q_m)$
Step 2 *Decomposable Disjunction* ▷ Probabilistically independent disjuncts
  if $m > 1$ and $Q_{\mathsf{UCNF}} = Q_1 \vee Q_2$ where $Q_1 \perp Q_2$ **then**
    $q_1 \leftarrow \mathbf{L}(Q_1, \mathcal{P}_{|_{Q_1}})$ and $q_2 \leftarrow \mathbf{L}(Q_2, \mathcal{P}_{|_{Q_2}})$
    return $1 - (1 - q_1) \cdot (1 - q_2)$
Step 3 *Inclusion-Exclusion*
  Apply cancellations/minimizations on Q.
  if $m > 1$ but Q_{UCNF} has no independent sub-query Q_i **then**
    return $\sum_{s \subseteq m} (-1)^{|s|+1} \cdot \mathbf{L}(\bigwedge_{i \in s} Q_i, \mathcal{P}_{|_{\wedge_{i \in s} Q_i}})$
Step 4 *Decomposable Conjunction* ▷ Probabilistically independent conjuncts
  Convert Q back to CNF: $Q_{\mathsf{CNF}} \coloneqq \forall \vec{x}\, Q_1 \wedge \ldots \wedge Q_k$
  if $Q = Q_1 \wedge Q_2$ where $Q_1 \perp Q_2$ **then**
    return $(\mathbf{L}(Q_1, \mathcal{P}_{|_{Q_1}}) \cdot \mathbf{L}(Q_2, \mathcal{P}_{|_{Q_2}}))$
Step 5 *Decomposable Universal Quantifier* ▷ Prob. independent projection
  if Q has a *separator variable* x **then**
    let T be all constants as x*-argument* in $\mathcal{P}$
    $q_c \leftarrow \prod_{t \in T} \mathbf{L}(Q[x/t], \mathcal{P}_{|_{x=t}})$ ▷ Ground and recurse over known atoms
    $q_o \leftarrow \mathbf{L}(Q[x/t], \emptyset)$ for some $t \in \Delta \setminus T$ ▷ Recurse over a canonical open atom
    return $q_c \cdot q_o^{|\Delta \setminus T|}$ ▷ Generalize the computation to the size of the domain
Step 6 *Fail*

Step 1. The *first step* is to rewrite the query Q into a union (disjunction) of CNF sentences, called union CNF, or UCNF. For example, the CNF formula

$$(\mathsf{R}(x) \vee \mathsf{S}(y, z)) \wedge (\mathsf{S}(x, y) \vee \mathsf{T}(x)),$$

can be rewritten as the disjunction of the CNF formulas, given as

$$\mathsf{R}(x) \wedge \mathsf{S}(x, y) \quad \text{and} \quad \mathsf{R}(x) \wedge \mathsf{T}(x) \quad \text{and} \quad \mathsf{S}(y, z) \wedge \mathsf{S}(x, y) \quad \text{and} \quad \mathsf{S}(y, z) \wedge \mathsf{T}(x).$$

The intuition behind this transformation is to produce multiple disjuncts from the given CNF in order to make (disjunctive) independencies explicit (if there are any). Note that such a rewriting does not always produce multiple disjuncts, in which case the formula is clearly also a CNF.

Step 2. The *second step* applies when the resulting UCNF has multiple disjuncts (or equivalently if it is not a CNF). The algorithm checks whether it is possible to partition the query into two UCNF formulas such that $Q := Q_1 \vee Q_2$, where Q_1 and Q_2 do not share any relational symbols, denoted $Q_1 \perp Q_2$, which ensures independence of Q_1 and Q_2. Then, it applies the probabilistic decomposition rule for disjunction:

$$\mathrm{P}(Q) = 1 - (1 - \mathrm{P}(Q_1)) \cdot (1 - \mathrm{P}(Q_2)).$$

It is easy to verify the correctness of this decomposition provided that Q_1 and Q_2 are independent terms and the latter is ensured by the fact that they do not share any relation symbols. The main idea in the second step (as well as in the remaining steps) is to recurse on simplified queries, using standard simplification rules of probability. Importantly, in the various recursions, the algorithm shrinks the set of atoms in the given PDB $\mathcal{P}$. Specifically, $\mathcal{P}_{|_Q}$ denotes the subset of $\mathcal{P}$ that only contains atoms for the predicates that appear in Q.

Step 3. The *third step* also applies only when the UCNF has multiple disjuncts and recurses using the *inclusion-exclusion* principle:

$$\mathrm{P}(Q) = \sum_{s \subseteq m} (-1)^{|s|+1} \cdot \mathrm{P}(\wedge_{i \in s} Q_i).$$

The key aspect in this step is to apply cancellations before the inclusion-exclusion step. The idea is to remove redundancies from the query and minimize it by checking for CNF formulas that are implied by others. This can be done using standard algorithms (Sagiv and Yannakakis 1980). An alternative is to use the *Möbius function* to detect the term to be cancelled as in (Dalvi and Suciu 2012). Either way, it is important to note that these manipulations are only on the query and therefore are independent of the database.

Step 4. In the *fourth step* the query is rewritten back as a CNF. Then the algorithm checks for independent sets of clauses in the CNF such that $Q = Q_1 \wedge Q_2$, where Q_1 and Q_2 do not share any relational symbols. If this is the case, then it applies the probabilistic decomposition rule for conjunction

$$\mathrm{P}(Q) = \mathrm{P}(Q_1) \cdot \mathrm{P}(Q_2).$$

Step 5. The *fifth step* is the workhorse of $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$, and the key difference with the $\mathsf{Lift}^{\mathsf{R}}$ algorithm of (Gribkoff, Van den Broeck, and Suciu 2014b). 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 $Q[x/t_1]$ and $Q[x/t_2]$ are independent. Hence, by multiplying $\overline{\mathrm{P}}(Q[x/t])$ for all t in the domain Δ, we obtain $\overline{\mathrm{P}}(Q)$.

The implementation of step five in $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$ 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 $\Delta \setminus T$.

For (i), $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$ 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 $\Delta \setminus T$ 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.

Step 6. Finally, $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$ can fail in *step six*, yielding no answer. We will discuss the meaning of this step in the next section.

4.2.3 Data Complexity Results

We now study the data complexity of upper and lower probabilistic query evaluation. We start our analysis with unions of conjunctive queries, and discuss the implications of Algorithm 1, in detail. Afterwards, we extend our results to other database queries.

Results for Unions of Conjunctive Queries

Using the terminology from (Dalvi and Suciu 2012), we say that queries are *safe* if the associated decision problem is in P, and *unsafe*, otherwise. The dichotomy of (Dalvi and Suciu 2012) is supported by an algorithm similar to $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$. When this algorithm fails, then the query is #P-hard. When it does not fail, it runs in P. This dichotomy-supporting algorithm has one major difference compared to $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$, 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 (whereas Suciu's algorithm implements this with the so-called Möbius function). This is a key aspect of the algorithm that ensures efficiency for all safe queries. Based on Theorem 4.17 and Corollary 3.18 we can lift the dichotomy result for UCQ queries in PDBs to OpenPDBs.

Corollary 4.21 (dichotomy) $\overline{\mathsf{P}}\mathsf{QE}(\mathsf{UCQ})$ *and* $\underline{\mathsf{P}}\mathsf{QE}(\mathsf{UCQ})$ *are either in* P*, or it is* PP*-complete for OpenPDBs in data complexity. Moreover, a* UCQ *is safe in OpenPDBs if and only if it is safe in PDBs.*

We already emphasized that the reduction given in Theorem 4.17 for the class of safe queries can be inefficient in practical terms as the construction results in a polynomial blow-up. Similarly; the preprocessing step in $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$ algorithm, can be polynomial. Ignoring the preprocessing, we show that, $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$ extended with cancellations in the inclusion-exclusion step, runs in linear time, if we also assume unit arithmetic cost, that is, the complexity of all arithmetic operations in the algorithm is fixed.

Theorem 4.22 *Let Q be a safe* UCQ *and $\mathcal{G}$ an OpenPDB. Then,* $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$ *computes* $\overline{\mathrm{P}}_{\mathcal{G}}(Q)$ (*respectively* $\underline{\mathrm{P}}_{\mathcal{G}}(Q)$) *in time polynomial in data complexity. Furthermore, if we assume unit arithmetic cost, then all safe queries can be evaluated in* linear time *(on the preprocessed OpenPDB) in data complexity.*

Proof. First, we show that the number of calls in the recursion tree of Algorithm 1 is linear in the size of $\mathcal{P}$ (i.e., the number of database atoms). We can ignore calls below each invocation of Line 23, as these calls no longer depend on $\mathcal{P}$. For the remaining

calls to $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$, we show a constant upper bound on how many calls are added when a single tuple t is added to $\mathcal{P}$. We say that a $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$-call *covers* an atom if that atom appears in its $\mathcal{P}$-argument. In Step 5, the separator variable must appear in every atom, which means that the separator variable must appear in t as well. Hence, of the child calls generated in Line 22, at most one can cover t. The number of calls that cover t is therefore bounded above by the number of recursive calls that can be generated in Steps 2–4. These steps are independent of $\mathcal{P}$, and only a function of the query. Therefore, the number of calls covering t is at most a constant in the size of $\mathcal{P}$. Every call to $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$ must cover at least one atom (ignoring the constant cost of Line 23 and its calls with empty databases), which bounds the number of calls to be linear in $\mathcal{P}$. Second, we show that the computations inside each individual call to $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$ admit an overall linear complexity. When adding an atom t to $\mathcal{P}$, the calls that cover t are of two types: (1) calls that do not cover another atom in $\mathcal{P}$ except for t, and (2) calls that already cover another atom in $\mathcal{P}$.

(1) Because of database restriction operators such as $\mathcal{P}_{|x=c}$ throughout Algorithm 1, the calls of type 1 all have the minimum required database as an argument, that is, $|\mathcal{P}| = 1$. Thus we have a constant data complexity for non-recursive computations in calls of type 1.

(2) To analyze the complexity of type-2 calls, we make the standard assumption that the domain has a given one-to-one correspondence with the integers. This allows for the operation $\mathcal{P}_{|x=c}$ to be implemented in linear time for all c simultaneously (i.e., Step 5 has linear data complexity sans the recursive calls). The complexity of type-2 calls thus grows linearly with $\mathcal{P}$.

In total, adding an atom to $\mathcal{P}$ will add a constant number of type-1 calls, whose internal computations all have constant data complexity. It will also increase the runtime of a constant number of type-2 calls by a constant. This gives an overall linear data complexity. □

This, in particular, implies that the algorithm $\mathsf{Lift}^{\mathsf{R}}$, which is a special case of $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$, also runs in linear time. Besides, we already noted that the preprocessing can be polynomial in OpenPDBs, while it remains linear in PDBs. Overall, these imply a stronger dichotomy for PDBs.

Corollary 4.23 $\mathsf{PQE}(\mathsf{UCQ})$ *for PDBs is either in linear time, or it is* PP*-complete if we assume* fixed precision *for the probability values in the PDB.*

Note that we also relaxed the unit arithmetic cost assumption: for PDBs, it is sufficient to assume that the precision of the probability values do not grow arbitrarily. This is needed to fix the cost of the multiplication, for which the best time bounds known are above linear time.

Corollary 4.23 extends the dichotomy of (Dalvi and Suciu 2012) from polynomial time to linear time if we assume fixed precision for probability values. 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 2016), do not apply to unions of conjunctive queries, nor to the dichotomy-supporting algorithm of (Dalvi and Suciu 2012).

Results Beyond Unions of Conjunctive Queries

Let us first start by a simple observation for OpenPDBs, which is very useful in the remaining of this chapter. By Lemma 4.18, we know that $\overline{\mathrm{P}}(Q) = 1 - \underline{\mathrm{P}}(\neg Q)$. In particular, this implies that

$$\overline{\mathrm{P}}_{\mathcal{G}}(Q) > p \text{ if and only if it is } \textit{not} \text{ the case that } \underline{\mathrm{P}}_{\mathcal{G}}(\neg Q) \geq 1 - p,$$
$$\underline{\mathrm{P}}_{\mathcal{G}}(Q) > p \text{ if and only if it is } \textit{not} \text{ the case that } \overline{\mathrm{P}}_{\mathcal{G}}(\neg Q) \geq 1 - p.$$

for any OpenPDB $\mathcal{G}$, query Q and threshold value p. Besides, since $\forall\mathsf{FO}$ and $\exists\mathsf{FO}$ queries are dual to each other, the probabilistic query evaluation for these queries can be reduced to each other by taking the complement of the respective problem. In essence, all complexity results for the problem $\overline{\mathsf{PQE}}(\forall\mathsf{FO})$, obtained in this section immediately hold for the complement of the problem $\underline{\mathsf{P}}\mathsf{QE}(\exists\mathsf{FO})$, and vice versa. Similarly, all complexity results for the problem $\underline{\mathsf{P}}\mathsf{QE}(\forall\mathsf{FO})$ hold for the complement of the problem $\overline{\mathsf{PQE}}(\exists\mathsf{FO})$, and vice versa. We refer this as the *duality property* and use it to simplify the proofs of some of the theorems. Practically speaking, this allows us to state the results regarding both to $\exists\mathsf{FO}$ and $\forall\mathsf{FO}$ queries, while providing the proof details only for one of these classes.

For unions of conjunctive queries, we have seen that the data complexity dichotomy in PDBs can be lifted to OpenPDBs: all safe queries remain safe. This was possible, because the satisfaction relation for unions of conjunctive queries is monotone. How does the picture look like for more expressive queries, where the monotone satisfaction relation does not hold any more? Obviously, there is no easy way of determining the completion that maximizes or minimizes the query probability as before.

Therefore, the striving questions are the following ones: can a safe query become hard for $\forall\mathsf{FO}$ and $\exists\mathsf{FO}$ queries? The next theorem answers this question using some involved techniques.

Theorem 4.24 *There exists a $\forall\mathsf{FO}$ query Q, for which $\mathsf{PQE}(Q)$ is in polynomial time for PDBs, whereas $\overline{\mathsf{PQE}}(Q)$ is* NP*-complete for OpenPDBs. There exists a $\exists\mathsf{FO}$ query Q, for which $\mathsf{PQE}(Q)$ is in polynomial time for PDBs, whereas $\underline{\mathsf{P}}\mathsf{QE}(Q)$ is* coNP*-complete for OpenPDBs.*

Proof. We prove the result for universal queries, which, by the duality property, entails the result for existential queries. The proof is composed of several steps, all based on techniques that may be interesting on their own right. To simplify the presentation, we therefore first provide an overview of the proof and the details follow afterwards.

First, we consider a $\forall\mathsf{FO}$ query, called Q_{SAT} that is known to be *unsafe*. Second, we transform the query Q_{SAT} into a query Q_{EQ} that consists of individually *safe* clauses (although Q_{EQ} itself remains unsafe). Moreover, our transformation ensures that the queries Q_{SAT} and Q_{EQ} are equisatisfiable. Even more is actually true: there is a one-to-one mapping between the models of these queries.

Our final transformation on the query applies an interesting technique to produce a safe query from Q_{EQ}. Recall that Q_{EQ} consists of only safe clauses and yet it is not safe. The main reason is that the terms produced in the inclusion-exclusion step are hard to evaluate. Put in more intuitive terms, clauses in Q_{EQ} are probabilistically dependent of each other, and this serves as the source for hardness. We manipulate these clauses so

that they become independent, which in turn helps us to come up with a query, called Q_{SAFE}, that is safe for PDBs. We then prove that Q_{SAFE} is indeed safe for PDBs.

Claim 1. The $\forall\mathsf{FO}$ query Q_{SAFE} is *safe* for PDBs.

Q_{SAFE} is defined in such a way that all the unsafe terms will cancel out during the exhaustive application of the inclusion-exclusion rule; that is the main reason why Q_{SAFE} is safe for PDBs. We finally show that this is not the case for OpenPDBs, by providing a reduction from satisfiability of propositional $\mathsf{3CNF}$ formulas.

Claim 2. The $\forall\mathsf{FO}$ query Q_{SAT} is NP-hard for OpenPDBs.

This implies that a query that is safe for PDBs is NP-hard for OpenPDBs. Therefore, it only remains to show that Q_{SAT} is in NP for OpenPDBs, which is immediate: for any OpenPDB and query Q, we can simply guess a completion $\mathcal{P}$ and check whether $\mathrm{P}_{\mathcal{P}}(Q) > p$ and $\mathcal{D} \models Q$ in polynomial time in data complexity using a nondeterministic Turing machine. This concludes the overview of the proof. We now provide the details of the individual claims and their corresponding constructions.

From $\mathbf{Q_{SAT}}$ to $\mathbf{Q_{EQ}}$. We show how to transform Q_{SAT} into an equisatisfiable query Q_{EQ}, using a special type of Tseitin transformation (Tseitin 1968). The idea is to detect the unsafe fragments in each clause of Q_{SAT} and replace them recursively with fresh atoms until the clause is safe. While doing so, we also add additional clauses to the formula, which assert the equivalence of the freshly introduced atom to the old formula, ensuring the overall equisatisfiability of Q_{SAT} and Q_{EQ}.

We omit the full details of this transformation (as it is well-known), but explain it on a small example. Consider, for instance a clause $\forall x, y\, \mathsf{L}(x) \vee \mathsf{L}(y) \vee \mathsf{R}_1(x, y)$, which is not safe as there is no separator variable. To transform this query, we define a fresh atom $\mathsf{Z}(x, y)$ to be equivalent to the formula $\mathsf{L}(y) \vee \mathsf{R}_1(x, y)$, which results in the following query:

$$\forall x, y\, (\mathsf{L}(x) \vee \mathsf{Z}(x, y)) \wedge (\mathsf{Z}(x, y) \leftrightarrow (\mathsf{L}(y) \vee \mathsf{R}_1(x, y))).$$

Notice that the first conjunct is already safe. The second conjunct in the query can further be simplified as

$$\begin{aligned}&\forall x, y\, (\ \ \mathsf{Z}(x, y) \rightarrow (\mathsf{L}(y) \vee \mathsf{R}_1(x, y))) \wedge ((\ \ \mathsf{L}(y) \vee \ \ \mathsf{R}_1(x, y)) \rightarrow \mathsf{Z}(x, y)) \equiv \\ &\forall x, y\, (\neg\mathsf{Z}(x, y) \vee \ \ \ \mathsf{L}(y) \vee \mathsf{R}_1(x, y)\ ) \wedge ((\neg\mathsf{L}(y) \wedge \neg\mathsf{R}_1(x, y)) \vee \ \mathsf{Z}(x, y)).\end{aligned}$$

Note that the first clause is also safe as there y is a separator variable (and afterwards x serves as a separator variable). However, the last clause is not in disjunctive form but can be decomposed into two disjunctive clauses

$$(\neg\mathsf{L}(y) \vee \mathsf{Z}(x, y)) \wedge (\mathsf{Z}(x, y) \vee \neg\mathsf{R}_1(x, y)),$$

both of which are safe. Clearly, we can apply this transformation to any universally quantified formula, and it will eventually result in a (potentially) large conjunction of clauses, each of which is individually safe. As a consequence, we obtain a query $Q_{\mathsf{EQ}} \coloneqq \bigwedge q_i(x, y, z)$, where each $q_i(x, y, z)$ is a safe clause over the variables x, y and z. By construction, it is easy to see that any model of Q_{SAT} can be extended to a model of Q_{EQ}, by interpreting the freshly introduced atoms to be equivalent to the sub-formula they replaced.

From $\mathbf{Q}_{\mathsf{EQ}}$ to $\mathbf{Q}_{\mathsf{SAFE}}$. We have already observed that every clause $q_i(x, y, z)$ in the query $Q_{\mathsf{EQ}} \coloneqq \bigwedge_{1 \le i \le n} q_i(x, y, z)$ is safe, while the query itself is still not safe. As noted before, the reason is hidden in the inclusion-exclusion terms that are hard to evaluate. We define the following formula

$$\begin{aligned} Q_{\mathsf{SAFE}} \coloneqq\ \forall x, y, z \bigwedge_{1 \le i \le n} (q_i \vee \neg \mathsf{H}_i) \wedge \bigwedge_{1 < i \le n} (\neg \mathsf{H}_1 \vee \neg \mathsf{H}_i) \\ \bigwedge_{2 < i \le n} (\neg \mathsf{H}_2 \vee \neg \mathsf{H}_i) \\ \cdots \\ \wedge\ (\neg \mathsf{H}_{n-1} \vee \neg \mathsf{H}_n) \\ \wedge\ (\bigvee_{1 \le i \le n} \mathsf{H}_i), \end{aligned}$$

where H_i is a zero-arity predicate. Let us give some insight on this formula. Note that the second part of the formula consists of only H_i-atoms. The last clause simply says that at least one of the atoms H_i must be true. Together with the other clauses, the second part of the formula asserts that exactly one atom H_k, for some $1 \le k \le n$ can be true, and all the remaining atoms H_i, $1 \le i \ne k \le n$ must be false. This has direct implications on the first part of the formula where the clauses q_i from the original formula Q_{EQ} appear. Briefly stated, if we choose to satisfy the k-th atom, H_k, then this means all clauses $q_i \vee \neg \mathsf{H}_i$ will be trivially satisfied for $i \ne k$. Intuitively, this means that their influence on the query probability will be fixed, and only $q_k \vee \neg \mathsf{H}_k$ will be counted. We are now ready to prove the claim.

Observe that the clauses in Q_{SAFE} of the form $\neg \mathsf{H}_i \vee q_i$ consist of multiple atoms that are not connected by a relational variable. This makes the clauses independent. Therefore, the entire query can be written as $(\neg \mathsf{H}_i \wedge \Delta) \vee (q_i \wedge \Delta)$, which is a UCNF. Applying this transformation to all clauses (all such clauses have disconnected H_i-atoms), we obtain a large union, on which we can perform inclusion-exclusion in Step 3. The detailed implementation of inclusion-exclusion for PDBs (cf. (Gribkoff, Van den Broeck, and Suciu 2014b)) removes a large number of unsatisfiable CNF clauses from this union. Afterwards, all remaining CNF formulas in the union have the form

$$\beta_k = \neg \mathsf{H}_1 \wedge \cdots \wedge \mathsf{H}_k \wedge q_k \wedge \ldots \neg \mathsf{H}_n,$$

that is, one H_i-atom for every i, and containing exactly one positive atom H_k with a corresponding clause $q_k(x, y, z)$. The entire UCNF is then given by $\bigvee_{i=1}^{n} \beta_i$. Importantly, note that the individual formulas β_i are mutually exclusive, removing the need for any nonsingular combination of β_i-terms in the inclusion-exclusion formula. Thus, the inclusion-exclusion rule computes

$$\mathrm{P}_{\mathcal{P}}(Q_{\mathsf{SAFE}}) = \sum_{i=1}^{m} \mathrm{P}_{\mathcal{P}}(\beta_i),$$

for some arbitrary PDB $\mathcal{P}$. Then, since q_i and H-atoms do not share any relation name, we can further decompose the query as

$$\mathrm{P}_{\mathcal{P}}(\beta_i) = \mathrm{P}_{\mathcal{P}}(q_i) \cdot \mathrm{P}_{\mathcal{P}}(\neg \mathsf{H}_1 \wedge \cdots \wedge \mathsf{H}_i \wedge \ldots \neg \mathsf{H}_n)$$

where the first term of the multiplication is safe by construction and it is easy to see that the second term is also safe. Thus, we obtain

$$\mathrm{P}_{\mathcal{P}}(Q_{\mathsf{SAFE}}) = \sum_{i=1}^{m} \mathrm{P}_{\mathcal{P}}(q_i) \cdot \mathrm{P}_{\mathcal{P}}(\neg \mathsf{H}_1 \wedge \cdots \wedge \mathsf{H}_i \wedge \ldots \neg \mathsf{H}_n),$$

which allows us to conclude that Q_{SAFE} is safe for PDBs and thus the proof of Claim 1.

What remains is to prove the Claim 2. We define an OpenPDB $\mathcal{G}_\varphi = (\mathcal{P}_\varphi, 1)$ over a vocabulary σ that contains the relation symbols from Q_{SAFE} (thus also from Q_{EQ}) as well as some additional constants (while assuming at least 3 of them). Intuitively Q_{SAFE} encodes (in a loose sense) the satisfaction conditions of a given 3CNF formula φ. We also define the PDB $\mathcal{P}_\varphi$ that stores the structure of φ as follows.

- $\mathcal{P}_\varphi$ contains all atoms $\langle \mathsf{H}_i : 0.5 \rangle$ for $1 \leq i \leq n$.
- The clauses φ_i are described with the help of the predicates R_1, …, R_4, each of which corresponds to one type of clause (as in the constructions in Chapter 3). All other R-atoms that do not correspond in such a way to one of the clauses, we add with probability 1 to $\mathcal{P}_\varphi$.
- All the remaining atoms (that are directly related to the satisfaction of Q_{EQ}) are left open. In other words, there are only n probabilistic atoms.

Based on these constructions, we now prove that the formula Φ is satisfiable if and only if $\overline{\mathrm{P}}_{\mathcal{G}_\varphi}(Q_{\mathsf{SAFE}}) \geq n \cdot (0.5)^n$. Let us assume that $\overline{\mathrm{P}}_{\mathcal{G}_\varphi}(Q_{\mathsf{SAFE}}) \geq n \cdot (0.5)^n$. This implies that there exists a completion $\mathcal{P}_\lambda$ that sets a choice for the open atoms such that $\mathrm{P}_{\mathcal{P}_\lambda}(Q_{\mathsf{SAFE}}) \geq n \cdot (0.5)^n$. By the structure of Q_{SAFE}, we already know that there are n different configurations of the H_i-atoms, each with probability $(0.5)^n$. This means that all $q_i(x, y, z)$, $1 \leq i \leq n$ must be satisfied by a distinct database with probability $(0.5)^n$. Note, however, all these databases differ only with respect to the H_i-atoms. Apart from H-atoms, all the atoms are deterministic in the completion; that is, they either are in the completion with probability 1, or are excluded. This implies that there exists a database $\mathcal{D}$ that satisfies all $q_i(x, y, z)$, $1 \leq i \leq n$. We now define a propositional assignment τ such that it maps a variable u in φ to true if and only if $\mathsf{L}(u) \in \mathcal{D}$. It is then easy to show that $\tau \models \varphi$.

Conversely, let us assume that φ is satisfiable and let τ be a satisfying assignment of φ. We define the completion $\mathcal{P}_\tau$ as follows. We add all atoms $\langle \mathsf{H}_i : 0.5 \rangle$ for $1 \leq i \leq n$ to the completion $\mathcal{P}_\tau$. Moreover, we add $\langle \mathsf{L}(u) : 1 \rangle$ if τ maps u to true; otherwise, we add $\langle \mathsf{L}(u) : 0 \rangle$ to the completion $\mathcal{P}_\tau$. Notice that all Z-atoms are introduced to be equivalent to some L-atom (in the construction of the query). To preserve this, we also add the respective Z-atoms, either with probability 0, or 1, depending on the L-atoms.

As before, there are n configurations of H_i-atoms, each with 0.5^n probability. For each such configuration exactly one q_i must be satisfied, which holds since each database $\mathcal{D}$ induced by $\mathcal{P}_\tau$ differs only on the H-atoms and it is easy to verify that each of them satisfies $\mathcal{D} \models q_i$ for all $1 \leq i \leq n$. Thus, we obtain that $\mathrm{P}_{\mathcal{P}_\lambda}(Q_{\mathsf{SAFE}}) \geq \mathrm{P}_{\mathcal{P}_\tau}(Q_{\mathsf{SAFE}}) = n \cdot (0.5)^n$, which concludes the proof of the claim and the result. □

Therefore, we conclude that a safe ∀FO (respectively ∃FO) query in PDBs can become NP-hard (respectively, coNP-hard) in OpenPDBs. We can then extend our treatment

to unsafe queries: can a query that is PP-complete in PDBs become harder for ∀FO and ∃FO queries in OpenPDBs? The answer is again positive.

Theorem 4.25 $\overline{\mathsf{PQE}}(\forall\mathsf{FO})$ *is* $\mathrm{NP}^{\mathrm{PP}}$*-complete and* $\underline{\mathsf{PQE}}(\exists\mathsf{FO})$ *is* $\mathrm{coNP}^{\mathrm{PP}}$*-complete for OpenPDBs in data complexity.*

Proof. We again only prove the result for universal queries, as the problems are dual. Let $\mathcal{G} = (\mathcal{P}, \lambda)$ be an OpenPDB, Q a ∀FO query and $p \in [0, 1)$. Consider a nondeterministic Turing machine with a PP oracle: first nondeterministically guess a λ-completion $\hat{\mathcal{P}}$ and then verify whether $\mathrm{P}_{\hat{\mathcal{P}}}(Q) > p$. This verification step is in PP as shown in Theorem 3.22, which proves membership.

As for the lower bound, we reduce the following problem. Let

$$\Phi \coloneqq \exists x_1, \ldots, x_\ell \, \mathsf{C}^c \, y_1, \ldots, y_m \; \varphi,$$

denote a quantified Boolean formula, where C represents the *counting quantifier* and $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_k$ is a propositional formula in 3CNF, defined over the variables $x_1, \ldots, x_\ell$, $y_1, \ldots, y_m$. Deciding the validity of such formulas is $\mathrm{NP}^{\mathrm{PP}}$-complete (Wagner 1986). Intuitively, this amounts to checking whether there is a partial assignment for $x_1, \ldots, x_\ell$ that admits at least c extensions to $y_1, \ldots, y_m$ that satisfy φ. To reduce the problem to upper probabilistic query evaluation, we consider again the universally quantified query Q_{SAT} given in the proof of Theorem 4.24. As before, Q_{SAT} is used to encode the satisfaction conditions of the formula Φ together with the PDB $\mathcal{P}_\Phi$ that stores the structure of Φ. The PDB $\mathcal{P}_\Phi$ is given as follows: for each variable y_j, $1 \leq j \leq m$, $\mathcal{P}_\Phi$ contains the atoms $\langle \mathsf{L}(y_j) : 0.5 \rangle$, where we view each y_j as a constant. As in the proof of Theorem 4.24, the clauses φ_i are described with the help of the predicates R_1, …, R_4. All other R-atoms that do not correspond in such a way to one of the clauses, we add with probability 1 to $\mathcal{P}_\Phi$. Finally, we define the OpenPDB $\mathcal{G}_\Phi = (\mathcal{P}_\Phi, 1)$. The construction provided for Q_{SAT} and $\mathcal{G}_\Phi$ is clearly polynomial. Furthermore, the query is fixed, and only $\mathcal{P}_\Phi$ depends on Φ. We now prove the following claim.

Claim. The formula Φ is valid if and only if $\overline{\mathrm{P}}_{\mathcal{G}_\Phi}(Q_{\mathsf{SAT}}) \geq \mathsf{c} \cdot (0.5)^m$.

Suppose that Φ is valid. Then, for some assignment μ of the variables $x_1, \ldots, x_\ell$, there are at least c different assignments τ extending μ to the variables $y_1, \ldots, y_m$ that satisfy Φ. We use the assignment μ in order to set a choice for all open atoms $\mathsf{L}(x_i)$, $1 \leq i \leq \ell$. More precisely, we define the λ-completion $\mathcal{P}_\mu$ that contains $\langle \mathsf{L}(x_i) : 1 \rangle$ if μ sets x_i to true and contains $\langle \mathsf{L}(x_i) : 0 \rangle$, otherwise. Intuitively, every assignment of the existentially quantified variables in Φ corresponds to a different completion and the assignment μ is realized by the completion $\mathcal{P}_\mu$.

Moreover, observe that for each satisfying assignment τ extending μ to the variables $y_1, \ldots, y_m$, there exists a database $\mathcal{D}$ induced by $\mathcal{P}_\mu$. We can define such a database $\mathcal{D}$ as follows: add all atoms to $\mathcal{D}$ that are in $\mathcal{P}_\mu$ with probability 1 and add every atom $\mathsf{L}(y_j)$ to $\mathcal{D}$ if and only if τ sets y_i to true. It is easy to see that each such database satisfies $\mathcal{D} \models Q_{\mathsf{SAT}}$. Finally, it suffices to observe that there are only m nondeterministic atoms in $\mathcal{P}_\mu$; namely the atoms $\mathsf{L}(y_j)$, $1 \leq j \leq m$ that correspond to the y-variables in Φ. Thus, every database $\mathcal{D}$ induced by $\mathcal{P}_\mu$ has the probability 0.5^m. By our assumption, there are c satisfying assignments τ extending μ; thus, it follows

that $\mathrm{P}_{\mathcal{P}_\mu}(Q_{\mathsf{SAT}}) = \mathsf{c} \cdot (0.5)^m$, which implies $\overline{\mathrm{P}}_{\mathcal{G}}(Q_{\mathsf{SAT}}) \geq \mathsf{c} \cdot (0.5)^m$ as a consequence of the query semantics in OpenPDBs.

For the other direction, let $\overline{\mathrm{P}}_{\mathcal{G}_\Phi}(Q_{\mathsf{SAT}}) \geq \mathsf{c} \cdot (0.5)^m$. Then, there exists a λ-completion $\mathcal{P}_\mu$ such that $\mathrm{P}_{\mathcal{P}_\mu}(Q_{\mathsf{SAT}}) \geq \mathsf{c} \cdot (0.5)^m$. Moreover, each database $\mathcal{D}$ induced by $\mathcal{P}_\mu$ sets a choice for the nondeterministic atoms $\mathsf{L}(y_1), \ldots, \mathsf{L}(y_m)$ and each such database has the probability $(0.5)^m$ (as there are only m nondeterministic atoms in the PDB). As a consequence, there must exist at least c databases induced by $\mathcal{P}_\mu$ that satisfies $\mathcal{D} \models Q$.

We define an assignment μ to the variables $x_1, \ldots, x_\ell$ such that x_i is mapped to true in μ if and only if $\langle \mathsf{L}(x_i) : 1 \rangle \in \mathcal{P}_\mu$. Then, for each database $\mathcal{D}$ induced by $\mathcal{P}_\mu$ and that satisfies $\mathcal{D} \models Q_{\mathsf{SAT}}$, we define an assignment τ that sets y_j to true if and only if $\mathsf{L}(y_j) \in \mathcal{D}$. It is then easy to verify that $\tau \models \Phi$ and that τ properly extends μ to a complete assignment. As there are c different assignments τ that extend μ while satisfying φ, we conclude that the formula Φ is valid. □

Nevertheless, we still have not clarified the complexity of evaluating the lower probabilities of universal queries (and upper probabilities of existential queries) for OpenPDBs. By reusing some ideas from Theorem 4.25, we show the following result.

Theorem 4.26 $\underline{\mathsf{PQE}}(\forall\mathsf{FO})$ *is* $\mathrm{coNP}^{\mathrm{PP}}$*-complete and* $\overline{\mathsf{PQE}}(\exists\mathsf{FO})$ *is* $\mathrm{NP}^{\mathrm{PP}}$*-complete for OpenPDBs in data complexity.*

Proof. We prove the first statement and the latter is entailed by duality. Let $\mathcal{G} = (\mathcal{P}, \lambda)$ be an OpenPDB, Q a $\forall\mathsf{FO}$ and $p \in [0, 1)$. Consider a nondeterministic Turing machine with access to a PP oracle, which is used to first guess a λ-completion $\mathcal{P}_\lambda$ and then to verify whether $\mathrm{P}_{\mathcal{P}_\lambda}(Q) \leq p$ using the PP oracle. The verification can be done in PP since the test $\mathrm{P}_{\mathcal{P}_\lambda}(Q) > p$ is in PP as shown before and PP is closed under complement. Finally, the Turing machine answers *no* if and only if the test for verification is positive. This proves membership.

We prove the lower bound is obtained in an analogous fashion to the proof of Theorem 4.25. We reduce the problem of deciding validity of formulas of the form

$$\Phi \coloneqq \forall x_1, \ldots, x_\ell \, \mathsf{C}^c \, y_1, \ldots, y_m \; \varphi,$$

which is dual to the problem we considered before. Moreover, we assume the exact same construction for $\mathcal{G} = (\mathcal{P}_\Phi, 1)$ and Q_{SAT} as in the proof of Theorem 4.25 with the only difference being that the x-variables are now universally quantified. We prove the following claim.

Claim. The formula Φ is valid if and only if $\underline{\mathrm{P}}_{\mathcal{G}_\Phi}(Q_{\mathsf{SAT}}) \geq \mathsf{c} \cdot (0.5)^m$.

Suppose that Φ is valid. Consider any λ-completion $\mathcal{P}_\lambda$ that sets a choice for the open atoms $\mathsf{L}(x_1), \ldots, \mathsf{L}(x_\ell)$. We define an instantiation μ such that μ maps x_i to true if and only if $\langle \mathsf{L}(x_i) : 1 \rangle$ is in the PDB $\mathcal{P}_\lambda$. Since Φ is valid, we know that for *any* instantiation of the variables $x_1, \ldots, x_\ell$, there exists at least c assignments τ that extends this instantiation to the variables $y_1, \ldots, y_m$ satisfying φ. Thus, there must exists at least c assignments τ extending μ such that $\tau \models \varphi$.

It is easy to see that each such assignment τ defines a database $\mathcal{D}_\tau$ induced by the PDB $\mathcal{P}_\lambda$ and that $\mathcal{D}_\tau \models Q_{\mathsf{SAT}}$. As before, every $\mathcal{D}_\tau$ has the probability 0.5^m since there

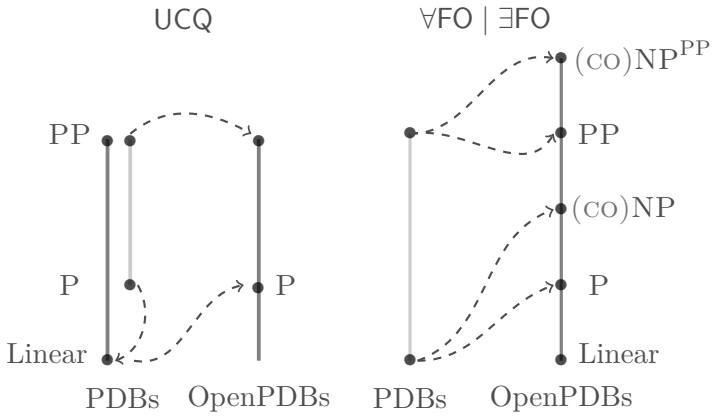


Figure 4.2: Complexity map for the data complexity results in OpenPDBs in comparison to PDBs .

are m nondeterministic atoms. This proves that for any completion the probability of the given query can not be less than $\mathsf{c} \cdot (0.5)^m$, which yields $\underline{\mathrm{P}}_{\mathcal{G}_\Phi}(Q_{\mathsf{SAT}}) \geq \mathsf{c} \cdot (0.5)^m$.

For the other direction, we know that regardless of the completion $\mathcal{P}_\lambda$ that is chosen, it holds that $\mathrm{P}_\lambda(Q_{\mathsf{SAT}}) \geq \mathsf{c} \cdot (0.5)^m$. Moreover, every completion corresponds to a valuation μ of the x-variables in Φ and for each such assignment can be extended to c satisfying assignments, as before.

Observe that every world induced by a completion $\mathcal{P}_\lambda$ has the probability 0.5^m. Thus, in order to satisfy $\mathrm{P}_{\mathcal{G}_\lambda}(Q_{\mathsf{SAT}}) \geq \mathsf{c} \cdot (0.5)^m$, there have to be c databases induced by $\mathcal{P}_\lambda$ satisfying Q_{SAT}. We have shown that each such database $\mathcal{D}_\tau$ corresponds to a satisfying assignment τ that extends μ such that $\tau \models \varphi$. Hence, there must be at least c such assignments. Finally, since we proved this for an arbitrary completion (hence, for an arbitrary valuation of the x-variables), we conclude that the Φ is valid. □

So far, the main results assert that upper probabilistic query evaluation is $\mathrm{NP}^{\mathrm{PP}}$-complete and lower probabilistic query evaluation is $\mathrm{coNP}^{\mathrm{PP}}$-complete in data complexity for both $\exists\mathsf{FO}$ and $\forall\mathsf{FO}$ queries. These upper bounds easily transfer to the case of first-order queries by similar arguments and we obtain our final result in data complexity.

Theorem 4.27 $\underline{\mathsf{PQE}}(\mathsf{FO})$ *is* $\mathrm{coNP}^{\mathrm{PP}}$*-complete and* $\overline{\mathsf{PQE}}(\mathsf{FO})$ *is* $\mathrm{NP}^{\mathrm{PP}}$*-complete for OpenPDBs in data complexity.*

Overview of the Data Complexity Results

Our results regarding the data complexity is summarized in Figure 4.2. First of all, we have shown that the data complexity dichotomy in PDBs for unions of conjunctive queries can be lifted to OpenPDBs. We also have presented an algorithm, $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$ that avoids polynomial blow-up whenever possible. By the properties of $\mathsf{Lift}^{\mathsf{R}}_{\mathsf{O}}$, we also observed that the original dichotomy of PDBs can be strengthened under mild assumptions. More precisely, by fixing the precision of the probabilities in the input PDB, all safe queries can be computed in linear time in PDBs (as depicted on the left-hand side of Figure 4.2). Unfortunately, this is not always the case for OpenPDBs.

We then extended our analysis to nonmonotone queries (see right hand side of Figure 4.2), which resulted in a richer complexity landscape. Once negation is allowed in the query atoms, it is not always possible to find the best completion in polynomial time. Instead, we may need to guess such a completion. This already shows up for safe queries in PDBs: they can remain safe or become (coNP-complete in OpenPDBs. A similar effect is observed for PP-complete queries, which can remain PP-complete, or become (coNP$^{\text{PP}}$-complete.

4.2.4 Combined Complexity Results

We also study the combined complexity of probabilistic query evaluation in OpenPDBs. We first look into bounded-arity combined complexity. Our first result is for unions of conjunctive queries, which coincides with the bounded-arity combined complexity results in PDBs.

Theorem 4.28 $\underline{\text{P}}\mathsf{QE}(\mathsf{UCQ})$ *and* $\overline{\text{P}}\mathsf{QE}(\mathsf{UCQ})$ *is* PP$^{\text{NP}}$*-complete for OpenPDBs in bounded-arity combined complexity.*

Proof. Let $\mathcal{G} = (\mathcal{P}, \lambda)$ be an OpenPDB, Q a UCQ and $p \in [0, 1)$. To show membership, we employ Theorem 4.16 and consider the extreme distributions by induced by the completions $\mathcal{P}$ and $\mathcal{P}_\lambda$, respectively. Since the arity of the predicates is bounded, the size of these completions is also bounded by a polynomial. Thus, we can reduce the upper (resp. lower) probabilistic query evaluation to probabilistic query evaluation in the PDB $\mathcal{P}_\lambda$ (resp. $\mathcal{P}$), which is in PP$^{\text{NP}}$ by Theorem 3.23. Hardness also follows from Theorem 3.23, which asserts that $\mathsf{PQE}(\mathsf{UCQ})$ is PP$^{\text{NP}}$-hard for PDBs. □

Interestingly, bounded-arity complexity appears to be the same as the data complexity for $\exists\mathsf{FO}$ and $\forall\mathsf{FO}$ queries. As shown in the next two theorems, this is mainly due to the power of the classes NP$^{\text{PP}}$ and coNP$^{\text{PP}}$, which do not gain additional computational power by another NP or coNP oracle.

Theorem 4.29 $\overline{\text{P}}\mathsf{QE}(\forall\mathsf{FO})$ *is* NP$^{\text{PP}}$*-complete and* $\underline{\text{P}}\mathsf{QE}(\exists\mathsf{FO})$ *is* coNP$^{\text{PP}}$*-complete for OpenPDBs in bounded-arity combined complexity.*

Proof. By the duality property, it is sufficient to prove the result for universal queries. Let $\mathcal{G} = (\mathcal{P}, \lambda)$ be an OpenPDB, Q a $\forall\mathsf{FO}$ query and $p \in [0, 1)$. Consider a nondeterministic Turing machine with a PP$^{\text{NP}}$ oracle: first nondeterministically guess a λ-completion $\hat{\mathcal{P}}$ (that is of size polynomial in bounded-arity complexity) and then verify whether $\text{P}_{\hat{\mathcal{P}}}(Q) > p$. This verification can be done using the PP$^{\text{NP}}$ oracle as shown in Theorem 3.24. This implies an upper bound NP$^{\mathfrak{C}}$, where $\mathfrak{C} = \text{PP}^{\text{NP}}$. Then, using Toda's result (Toda 1989), which asserts that PP$^{\text{PH}} \subseteq$ P$^{\text{PP}}$, it is easy to see that $\mathfrak{C} \subseteq$ P$^{\text{PP}}$ and thus NP$^{\mathfrak{C}} =$ NP$^{\text{PP}}$. This problem is clearly NP$^{\text{PP}}$-hard in bounded-arity combined complexity as is already in data complexity by Theorem 4.25. □

Theorem 4.30 $\underline{\text{P}}\mathsf{QE}(\forall\mathsf{FO})$ *is* coNP$^{\text{PP}}$*-complete and* $\overline{\text{P}}\mathsf{QE}(\exists\mathsf{FO})$ *is* NP$^{\text{PP}}$*-complete for OpenPDBs in bounded-arity combined complexity.*

Proof. We prove the result for universal queries. Let $\mathcal{G} = (\mathcal{P}, \lambda)$ be an OpenPDB, Q a $\forall\mathsf{FO}$ query and $p \in [0, 1)$. Consider a nondeterministic Turing machine with access to a

PP^{NP} oracle that is used to first guess a λ-completion $\hat{\mathcal{P}}$ (that is of size polynomial in bounded-arity complexity) and then to verify whether $\text{P}_{\hat{\mathcal{P}}}(Q) \leq p$ using the PP^{NP} oracle. The verification can be done in PP^{NP} by Theorem 3.24. Finally, the Turing machine answers *no* if and only if the test for verification is positive. This proves membership to $\text{coNP}^{\mathfrak{C}}$, where $\mathfrak{C} = \text{PP}^{\text{coNP}} = \text{PP}^{\text{NP}}$. Then, analogous arguments to those in the proof of Theorem 4.29 imply membership in coNP^{PP} based on Toda's result (Toda 1989). This problem is clearly coNP^{PP}-hard in bounded-arity complexity as is already in data complexity by Theorem 4.26. □

As for first-order queries, the complexity of classical query evaluation again dominates the complexity of probabilistic query evaluation in OpenPDBs.

Theorem 4.31 $\underline{\text{PQE}}(\text{FO})$ *and* $\overline{\text{PQE}}(\text{FO})$ *are* PSPACE*-complete for OpenPDBs in bounded-arity combined complexity.*

Proof. Let $\mathcal{G} = (\mathcal{P}, \lambda)$ be an OpenPDB, Q a FO query and $p \in [0, 1)$. Consider a *polynomial space bounded* nondeterministic Turing machine that enumerates exponentially many completions, each of which is of polynomial size. Then, for each of these completions $\hat{\mathcal{P}}$, it tests whether $\text{P}_{\hat{\mathcal{P}}}(Q) > p$, which is in PSPACE by Theorem 3.25. PSPACE-hardness follows from Theorem 3.25. □

The study of combined complexity is a somewhat intricate notion in OpenPDBs in the following sense: since neither the arity nor the schema is fixed, a completion, which is a PDB, can grow exponentially. Thus, we face a very high complexity. Furthermore, we still need to perform probabilistic inference over exponentially large completions. These highly intractable classes are beyond the focus of our work, but for the sake of completeness, we provide a simple upper bound.

Theorem 4.32 $\overline{\text{PQE}}(\text{FO})$ *and* $\underline{\text{PQE}}(\text{FO})$ *are both in* EXPSPACE *for OpenPDBs in combined complexity.*

Proof. Let $\mathcal{G} = (\mathcal{P}, \lambda)$ be an OpenPDB, Q an FO query and $p \in [0, 1)$. Consider an exponential space bounded Turing machine that enumerates all completions in exponential space. Then, for each of these completions $\hat{\mathcal{P}}$, it tests whether $\text{P}_{\hat{\mathcal{P}}}(Q) > p$. Since $\hat{P}$ is exponential size, this test is in EXPSPACE. □

This concludes our complexity analysis. Next, we give an overview of the results and discuss related work.

4.2.5 Overview of the Results, Outlook and Related Work

We observe that large scale knowledge bases are incomplete by their nature and argue this characteristic should be incorporated into the semantics. Our proposal is called OpenPDBs, where atoms that are not in the database still remain possible with some default probability. This is in contrast with PDBs, where atoms that do not appear in the database are assigned a probability 0 by the CWA.

Our work builds on the foundations of tuple-independent PDBs (Suciu et al. 2011), as surveyed in Chapter 3. In particular, we extend the dichotomy result of Dalvi and Suciu, for unions of conjunctive queries to OpenPDBs. As a side contribution, we observe

Table 4.3: Data, bounded-arity and combined complexity results for upper and lower (without parenthesis) probabilistic query evaluation in OpenPDBs.

Query Languages	Lower/Upper Probabilistic Query Evaluation in OpenPDBs		
	data	bounded-arity	combined
UCQ	P vs PP [Theorem 4.17] [Corollary 4.21]	PP^{NP} [Theorem 4.28]	in ExpSpace [Theorem 4.32]
∃FO	(co)NP^{PP} [Theorem 4.25]	(co)NP^{PP} [Theorem 4.29]	in ExpSpace [Theorem 4.32]
∀FO	(co)NP^{PP} [Theorem 4.26]	(co)NP^{PP} [Theorem 4.30]	in ExpSpace [Theorem 4.32]
FO	(co)NP^{PP} [Theorem 4.27]	PSpace [Theorem 4.31]	in ExpSpace [Theorem 4.32]

that the original dichotomy for PDBs is stronger: under reasonable assumptions all safe queries can be computed in linear time.

The results of our complexity analysis are summarized in Table 4.3, where the complexity of upper and lower probabilistic query evaluation can be distinguished by the parenthesis, i.e., lower probabilistic query evaluation is read without parenthesis. The complexity results align very well with probabilistic query evaluation in PDBs for unions of conjunctive queries (except for the combined complexity). Notably, probabilistic query evaluation is not harder in bounded-arity combined complexity for ∃FO and ∀FO queries. The precise relation between the dichotomy results is already emphasized in Figure 4.2.

A key challenge in OpenPDBs is to restrict the open-world to provide tighter probability bounds, which is an aspect to be revisited in Chapter 7, where we propose a specific approach to exclude spurious possible worlds, and limit the probability mass of open atoms. Our proposal extends OpenPDBs with an additional knowledge representation layer allowing us to obtain more informative probability bounds.

OpenPDBs are credal representations, and thus are also closely related to credal networks (Cozman 2000). Note that our complexity results align with that of credal networks which also show an increase from P to NP and from PP to NP^{PP} (De Campos and Cozman 2005) compared to Bayesian networks (Darwiche 2009). Besides, one source of hardness for probabilistic inference in credal networks is due to the conditional dependencies encoded in the network structure, which is very different from OpenPDBs as such dependencies stem from the query in OpenPDBs.

The high level motivation of this work is clearly linked to open-world reasoning. We note that the OWA is common for deterministic knowledge bases, mostly based on Datalog$^{\pm}$ and Description Logics and it is a driving force for these technologies over classical databases. A detailed overview of these formalisms is given in Chapter 6.

In OpenPDBs, 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, Marthi, Russell, Sontag, Ong, and Kolobov 2007), which is an interesting direction. We will revisit open-domain models in Part III in the context of ontology-mediated queries. To avoid explicit reasoning about all tuples individually

is the topic of *lifted inference* (Poole 2003). OpenPDBs bring together the high-level reasoning of lifted inference and the data-centric reasoning of probabilistic databases.

Chapter 5

Maximal Posterior Computations for Probabilistic Databases

Forming the foundations of large-scale knowledge bases, probabilistic databases have been widely studied in the literature. In particular, probabilistic query evaluation has been investigated intensively as a central inference mechanism. However, despite its power, query evaluation alone cannot extract all the relevant information encompassed in large-scale knowledge bases. To exploit this potential, we study two inference tasks; namely finding the *most probable database* and the *most probable hypothesis* for a given query, both of which are inspired by the *maximal posterior probability* computations in Probabilistic Graphical Models (PGMs) (Koller and Friedman 2009). We, therefore, first provide an overview on Bayesian Networks (Pearl 1988), an instance of PGMs, and then describe how similar computations can be useful in the context of probabilistic databases.

5.1 Probabilistic Graphical Models

Real-world domains are usually very complex and the parameters that need to be taken into consideration to model such domains can thus be relatively high. Consider a typical scenario from a medical domain, where a patient might have multiple possible diseases, might show different symptoms, and might possess several test results; that is to say, there may be many parameters that need to be taken into consideration while modeling such a domain. Notably, many of these parameters are probabilistically related to each other; for instance, having *high systolic pressure* may be related to having *high fever* with some probability. Note that this is in contrast with the strong independence assumption of PDBs.

These domains can be characterized in terms of a set of random variables, where each random variable defines a property of the domain we are interested in monitoring. In essence, we need to specify a joint distribution over the event space in order to be able to do inference over the model. On the other hand, specifying this joint probability distribution naïvely; for instance, by enumerating all possible worlds over the event space, is already a very demanding task in the computational sense. For 20 parameters, we need to enumerate 2^{20} many possible worlds, which leads to a state explosion, already in the modeling phase (before any inference is performed). The key insight in PGMs is to be able to represent a joint probability distribution in a relatively compact manner, by exploiting the structural properties in the probability distribution. The high-level idea is to encode the conditional independence assumptions into a graph structure, which typically results in an exponentially more succinct representation of the joint probability

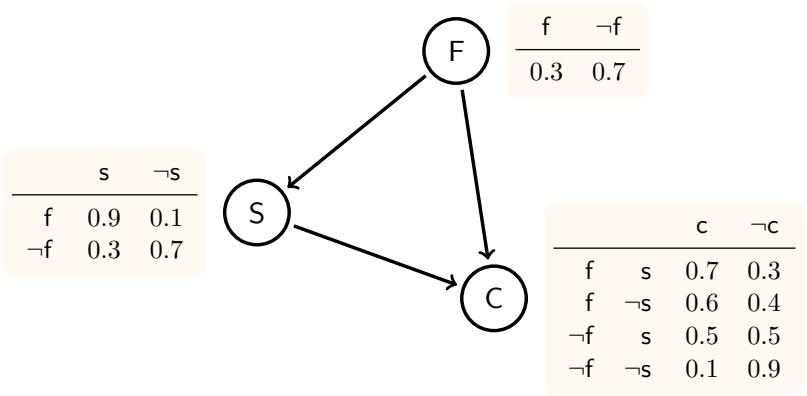


Figure 5.1: The BN $\mathcal{B}_h$ over the variables $V_h = \{\mathsf{F}, \mathsf{S}, \mathsf{C}\}$, where conditional probabilities are depicted using tables. Following notational conventions, we write x in place of $\mathsf{X} = 1$ and $\neg\mathsf{x}$ in place of $\mathsf{X} = 0$ to denote elementary events.

distribution. As an instance of PGMs, we focus on Bayesian Networks (Pearl 1988), which led to a breakthrough in artificial intelligence research; for other models, we refer the reader to (Koller and Friedman 2009).

5.1.1 Bayesian Networks

Bayesian networks (BNs) are probabilistic graphical models that compactly represent the *joint probability distribution* over a state space (Pearl 1988).

Definition 5.1 (Bayesian network) Let V be a finite set of random variables. A *Bayesian network (BN)* over V is a pair $\mathcal{B} = (G, \Theta)$, where $G = (V, E)$ is a directed acyclic graph, and Θ contains, for every random variable $X \in V$, a conditional probability distribution $\mathrm{P}_{\mathcal{B}}(X \mid \pi(X))$ for X, given its parents $\pi(X)$. ◊

The key insight in BNs is that the underlying directed acyclic graph encodes *conditional independence assumptions* in an effective manner. Put in Nilsson's words, every variable $X \in V$ in a BN is conditionally independent of its non-descendants, given its parents (Nilsson 1998). Every BN $\mathcal{B}$ thus defines the unique joint probability distribution:

$$P_{\mathcal{B}}(V) = \prod_{X \in V} P_{\mathcal{B}}(X \mid \pi(X))$$

over its variables V (see Definition 2.16). As before, a joint valuation ω of the variables V is called a *world*.

Definition 5.2 (probabilistic inference) Given a BN $\mathcal{B}$ over V, an event e over V, and a threshold value $p \in [0, 1)$, probabilistic inference in Bayesian networks is to decide whether $\mathrm{P}_{\mathcal{B}}(\mathsf{e}) > p$. ◊

In general, probabilistic inference in Bayesian networks is PP-complete (Littman, Majercik, and Pitassi 2001). Intuitively, the combinatorial aspect is related to the treewidth of the underlying graph. Thus, if the treewidth is small enough, inference becomes tractable.

Table 5.1: The joint probability distribution induced by the BN $\mathcal{B}_h$.

Worlds	F	S	C	$\mathrm{P}_{\mathcal{B}_h}(\omega_i)$
ω_0	×	×	×	.441
ω_1	×	×	✓	.049
ω_2	×	✓	×	.105
ω_3	×	✓	✓	.105
ω_4	✓	×	×	.012
ω_5	✓	×	✓	.018
ω_6	✓	✓	×	.081
ω_7	✓	✓	✓	.189

Example 5.3 Consider the BN $\mathcal{B}_h$ over the variables $V_h = \{\mathsf{F}, \mathsf{S}, \mathsf{C}\}$ depicted in Figure 5.1. It models a simplified health care assistance scenario by defining a joint probability distribution over the different states of a patient. As in Example 2.11, the variables F, S, and C denote *high fever*, *high systolic pressure*, and *high cholesterol*, respectively. Following conventional notations, we write f instead of $\mathsf{F} = 1$ and $\neg\mathsf{f}$ instead of $\mathsf{F} = 0$ and similarly for other random variables. The probability for the patient to have high systolic pressure is then given as

$$\begin{aligned}\mathrm{P}_{\mathcal{B}_h}(\mathsf{s}) &= \mathrm{P}_{\mathcal{B}_h}(\mathsf{s} \mid \mathsf{f}) \cdot \mathrm{P}_{\mathcal{B}_h}(\mathsf{f}) + \mathrm{P}_{\mathcal{B}_h}(\mathsf{s} \mid \neg\mathsf{f}) \cdot \mathrm{P}_{\mathcal{B}_h}(\neg\mathsf{f}) \\ &= .9 \cdot .3 + .3 \cdot .7 = .48.\end{aligned}$$

Conditional probabilities are specified in a compact way through the BN; thus, typically, the worlds (where the random variable evaluates to true) do not need to be enumerated to compute the probability. ◊

Importantly, the joint probability distribution that is defined by the BN $\mathcal{B}_h$ is exactly the same as the one given in Table 2.1 from Example 2.11. Differently, $\mathcal{B}_h$ is a more succinct representation.

5.1.2 Maximal Posterior Computations in Bayesian Networks

Computing the *maximal posterior probability* of a distinguished set of variables, given evidence about another set of variables, is one of the key computational problems in PGMs. In its general form, this problem is studied under the name of *maximum a posteriori hypothesis* (MAP), where the hypothesis refers to an instantiation of a set of distinguished variables. The *most probable explanation* (MPE) is a special case of MAP, where the distinguished variables and the evidence variables cover all variables in the model. The decision problem for MPE is defined as follows.

Definition 5.4 (MPE) Let $\mathcal{B}$ be a Bayesian network over V, $p \in [0, 1)$ a threshold value and e an event. MPE is to decide whether there exists an instantiation $\mathbf{v}$ of V such that $\mathrm{P}_{\mathcal{B}}(\mathbf{v}, \mathsf{e}) > p$. ◊

The corresponding function problem is to compute an instantiation $\mathbf{v}$ that maximizes the probability $\mathrm{P}_{\mathcal{B}}(\mathbf{v}, \mathsf{e}) > p$. Intuitively, this corresponds to finding a *world* with the *maximal* probability where the given event is *true*.

Table 5.2: The probabilistic database $\mathcal{P}_v$ encodes dietary regimes of some individuals, and the friendship relation among those individuals.

Vegetarian	
alice	0.7
bob	0.9
chris	0.6

FriendOf		
alice	bob	0.7
alice	chris	0.8
bob	chris	0.1

Eats		
bob	spinach	0.7
chris	mussels	0.8
alice	broccoli	0.2

Meat	
shrimp	0.7
mussels	0.9
seahorse	0.3

Example 5.5 Consider again the BN $\mathcal{B}_h$ and suppose that we are interested in the event s. It is easy to see that the instantiation f, c maximizes the objective function, as $\omega_7 = \{\mathsf{f}, \mathsf{c}, \mathsf{s}\}$ is the world with maximal probability where s holds. ◊

Formulated as a decision problem, MPE is NP-complete (Littman 1999; Shimony 1994). Note that MPE only allows for complete explanations, which may produce many redundancies in real applications. It is therefore more interesting to obtain partial explanations for events and, as opposed to MPE, this is possible for MAP.

Definition 5.6 (MAP) Let $\mathcal{B}$ be a Bayesian network over V, $p \in [0, 1)$ a threshold value, e an event and $X \subseteq V$ a set of distinguished variables. MAP is to decide whether there exists an instantiation $\mathbf{x}$ of the variables X such that $\mathrm{P}_{\mathcal{B}}(\mathbf{x}, e) > p$. ◊

The corresponding function problem is defined analogously. In this case, we are not necessarily interested in identifying a particular *world*, but a partial explanation, in the form of an event, also called *context*, with the *maximal* probability.

Example 5.7 Consider again the BN $\mathcal{B}_h$ and suppose that we are interested in the event s and let $\{\mathsf{F}\}$ be the singleton set that represents the set of distinguished variables. The task is to find an instantiation x of $\{\mathsf{F}\}$ which maximizes $\mathrm{P}_{\mathcal{B}_h}(\mathsf{x}, \mathsf{s})$. There are two possible instantiations with

$$\mathrm{P}_{\mathcal{B}_h}(\mathsf{f}, \mathsf{s}) = 0.27 > 0.21 = \mathrm{P}_{\mathcal{B}_h}(\neg\mathsf{f}, \mathsf{s}),$$

which means that the instantiation f needs to be chosen to maximize the posterior probability. ◊

MAP is $\mathrm{NP}^{\mathrm{PP}}$-complete (Park and Darwiche 2004b); that is, significantly harder than MPE under widely accepted complexity assumptions. Both for MPE and MAP, we need to compute an explanation, which is the intuitive reason behind the NP-hardness. The difference appears in the probability computation. The probability computation is easy for MPE, as we only need to compute the probability of a world. For MAP, we have to compute the probability of partial explanations, or equivalently events, which is PP-hard. For further details, we refer to (Darwiche 2009).

5.2 Explanations for Probabilistic Database Queries

Probabilistic query evaluation is the key inference task underpinning probabilistic knowledge bases. This is a natural problem and serves as the core task. Let us briefly revisit probabilistic query evaluation on the PDB given in Table 5.2.

Example 5.8 Consider the PDB $\mathcal{P}_v$ given in Table 5.2 and the conjunctive query

$$Q_{\mathsf{fr}} = \exists x, y\, Q_{\mathsf{veg}}(x) \wedge \mathsf{FriendOf}(x, y) \wedge Q_{\mathsf{veg}}(y),$$

through which we can ask the probability of vegetarians being friends with vegetarians. In the given PDB, alice, bob, and chris are all vegetarians and friends with each other (with some probability). The query

$$Q_{\mathsf{fr}}[x/\mathsf{bob}] = \exists y\; Q_{\mathsf{veg}}(\mathsf{bob}) \wedge \mathsf{FriendOf}(\mathsf{bob}, y) \wedge Q_{\mathsf{veg}}(y)$$

is a special case of Q_{fr} which asks whether bob has vegetarian friends. Its probability can be computed as $\mathrm{P}(Q_{\mathsf{fr}}[x/\mathsf{bob}]) = 0.9 \cdot 0.1 \cdot 0.6 = 0.054$. ◊

For this example, suppose we observe that $Q_{\mathsf{fr}}[x/\mathsf{bob}]$ is true and would like to learn what best explains this observation relative to the underlying probabilistic database. To be able to explain such an observation, we need different inference tasks than probabilistic query answering. We revisit this example after providing a principled approach for dealing with such tasks.

Inspired by the *maximal posterior probability* computations in PGMs, we investigate the problem of finding *most probable explanations* for probabilistic queries in order to exploit the potential of such large databases to their full extent. Both MPE and MAP translate to probabilistic databases in a natural way through the rich structure of queries and, as natural counterparts of the MPE and MAP problems in PGMs, these inferences can be used in a variety of applications that involve *prediction* or *diagnosis* tasks.

The *most probable database* problem (analogous to MPE), first proposed in (Gribkoff, Van den Broeck, and Suciu 2014a), is the problem of determining the (classical) database with the largest probability that satisfies a given query. Intuitively, the query defines constraints on the data, and the goal is to find the most probable database that satisfies these constraints. We also introduce a more intricate notion, called *most probable hypothesis*, which only asks for *partial* databases satisfying the query (analogous to MAP). The most probable hypothesis contains only atoms that contribute to the satisfaction condition of the query, which allows to more precisely pinpoint the most likely explanations of the query.

We study the computational complexity of the corresponding decision problems, denoted by MPD and MPH, respectively, for a variety of query languages, ranging from unions of conjunctive queries to first-order queries, and provide a detailed complexity analysis. Our results provide detailed insights about the nature of these problems.

5.2.1 The Most Probable Database Problem

The need for alternative inference mechanisms for probabilistic knowledge bases has been observed before, and the *most probable database* problem has been proposed in (Gribkoff, Van den Broeck, and Suciu 2014a).

Definition 5.9 Let $\mathcal{P}$ be a probabilistic database and Q a query. The *most probable database* for Q over $\mathcal{P}$ is given by

$$\arg\max_{\mathcal{D} \models Q} \mathrm{P}(\mathcal{D}),$$

where $\mathcal{D}$ ranges over all worlds induced by $\mathcal{P}$. ◊

Table 5.3: The most probable database for the query Q_{fr} over the PDB $\mathcal{P}_v$.

Vegetarian	
alice	0.7
bob	0.9
chris	0.6

FriendOf		
alice	bob	0.7
alice	chris	0.8
bob	chris	0.1

Eats		
bob	spinach	0.7
chris	mussels	0.8
alice	broccoli	0.2

Meat	
shrimp	0.7
mussels	0.9
seahorse	0.3

The intuition is very straight-forward: a probabilistic database defines a probability distribution over exponentially many classical databases, and the most probable database is the element in this collection that has the highest probability, while still satisfying the given query. This can be seen as the best instantiation of a probabilistic model, and hence analogous to MPE in BNs (Darwiche 2009).

Recall our running example and the given PDB $\mathcal{P}_v$. We can now filter the most probable database that satisfies the query

$$Q_{\mathsf{fr}} = \exists x, y \; Q_{\mathsf{veg}}(x) \wedge \mathsf{FriendOf}(x, y) \wedge Q_{\mathsf{veg}}(y).$$

Intuitively, the atoms $\mathsf{Vegetarian}(\mathsf{alice})$, $\mathsf{Vegetarian}(\mathsf{bob})$, $\mathsf{FriendOf}(\mathsf{alice}, \mathsf{bob})$ need to be included in the most probable database, as they satisfy the query with the highest probability (compared to other possible matches). Since the query is monotone, for the remaining atoms, we only need to decide whether including them results in a higher probability than excluding them, which amounts to deciding whether their probability is greater than 0.5. Table 5.3 shows the most probable database for Q_{fr} over the PDB $\mathcal{P}_v$, where all atoms that belong to the most probable database are highlighted.

We now show that identifying the most probable database can be more cumbersome if we consider other query languages; in particular, $\forall$FO queries.

Example 5.10 Consider again the PDB $\mathcal{P}_v$ and the query

$$Q_{\mathsf{veg}} = \forall x, y \; \neg\mathsf{Vegetarian}(x) \vee \neg\mathsf{Eats}(x, y) \vee \neg\mathsf{Meat}(y),$$

which defines the constraints of being a vegetarian, which is violated by the atoms $\mathsf{Vegetarian}(\mathsf{chris})$, $\mathsf{Eats}(\mathsf{chris}, \mathsf{mussels})$ and $\mathsf{Meat}(\mathsf{mussels})$. Therefore, the most probable database for Q_{veg} cannot contain all three of them, i.e., one of them has to be removed (from the explanation). In this case, it is easy to see that $\mathsf{Vegetarian}(\mathsf{chris})$ needs to be removed, as it has the lowest probability among them. Thus, the most probable database (in this case unique) contains all atoms of $\mathcal{P}_v$ that have a probability above 0.5, except for $\mathsf{Vegetarian}(\mathsf{chris})$.

Suppose now that we have observed $Q_{\mathsf{fr}}[x/\mathsf{bob}]$, and we are interested in finding an explanation for this observation under the constraint of Q_{veg}, which is specified by the query

$$Q_{\mathsf{vf}} = Q_{\mathsf{fr}}[x/\mathsf{bob}] \wedge Q_{\mathsf{veg}}.$$

Observe that $\mathsf{Vegetarian}(\mathsf{chris})$ and $\mathsf{FriendOf}(\mathsf{bob}, \mathsf{chris})$ must be in the explanation to satisfy $Q_{\mathsf{fr}}[x/\mathsf{bob}]$. Moreover, either $\mathsf{Eats}(\mathsf{chris}, \mathsf{mussels})$ or $\mathsf{Meat}(\mathsf{mussels})$ has to be excluded from the most probable database since otherwise Q_{veg} will be violated. The resulting most probable database is highlighted in Table 5.4. ◊

Table 5.4: The most probable database for the query Q_{vf} over the PDB $\mathcal{P}_v$.

Vegetarian	
alice	0.7
bob	0.9
chris	0.6

FriendOf		
alice	bob	0.7
alice	chris	0.8
bob	chris	0.1

Eats		
bob	spinach	0.7
chris	mussels	0.8
alice	broccoli	0.2

Meat	
shrimp	0.7
mussels	0.9
seahorse	0.3

We now formulate the most probable database as a decision problem, denoted by MPD, and investigate its computational complexity. Our formulation is analogous to the decision variants of maximal posterior computations in BNs.

Definition 5.11 (MPD) Let Q be a query, $\mathcal{P}$ a probabilistic database and $p \in (0, 1]$ a threshold. MPD is the problem of deciding whether there exists a database $\mathcal{D}$ that satisfies Q with $\mathrm{P}(\mathcal{D}) > p$. MPD is parametrized with a particular query language; thus, we write MPD(Q) to define MPD on the class Q of queries. ◊

As before, it is possible to reduce the test for $\geq$ to the test for $>$, and vice versa in the decision problem, which follows from analogous arguments to those given in Lemma 3.17.

Data complexity results for MPD

Our first result concerns the well-known class of *unions of conjunctive queries*: we show that MPD can be solved using at most *logarithmic space* and *polynomial time*. More precisely, our result asserts that it is possible to encode the MPD problem uniformly into a class of TC^0 circuits.

Theorem 5.12 MPD(UCQ) *is in* DLogTime*-uniform* TC^0 *in data complexity.*

Proof. Let $\mathcal{P}$ be a PDB, Q a UCQ and $p \in [0, 1)$ a threshold value. In principle, we can enumerate all the databases induced by the PDB $\mathcal{P}$ and decide whether there is a database that satisfies Q and has a probability greater or equal than p. This would require exponential time, as there are potentially exponentially many databases (worlds) induced by a probabilistic database. Fortunately, this can be avoided, as the satisfaction relation for union of conjunctive queries is monotone: once a database satisfies a UCQ, then any superset of this database will satisfy the UCQ. We can, therefore, design an algorithm, which initiates the database with the atoms resulting from a match for the query (where the match is over the atoms that appear with a positive probability in the PDB $\mathcal{P}$) and extends this to a database with maximal probability, as follows: it adds only those atoms from the PDB $\mathcal{P}$ to the database that appear with a probability higher than 0.5, ensuring to obtain the database with the maximal probability. As a consequence, all these databases satisfy Q and if, furthermore, there is a database $\mathcal{D}$ among them with $\mathrm{P}(\mathcal{D}) \geq p$, then the algorithm answers yes; otherwise, it answers no.

It is easy to see that this algorithm is correct. However, if performed naïvely, as described, it results in a polynomial blow-up: there are polynomially many matches for the query (in data complexity), and for each match, constructing a database requires polynomial time and space. Observe, on the other hand, that we can uniformly access every match through an AC^0 circuit, and we can avoid explicitly constructing a database for each such match. More concretely, since we are only interested in the probability of

the resulting database, we can perform an iterated multiplication over the probabilities of the individual atoms: if $\langle a : q \rangle \in \mathcal{P}$, where $q > 0.5$, we multiply with q, otherwise, we multiply with $1 - q$. Notice that we can perform such iterated multiplication with TC^0 circuits that can be constructed in DLogTime (Hesse, Allender, and Barrington 2002). Finally, it is sufficient to compare the resulting number with the threshold value p. This last comparison operation can also be done using uniform AC^0 circuits. This puts MPD(UCQ) in DLogTime-uniform TC^0 in data complexity. □

This low data complexity result triggers an obvious question: what are the sources of tractability? The answer is partially hidden behind the fact that the satisfaction relation for unions of conjunctive queries is monotone. This allows us to reduce the search space to polynomially many databases, which are obtained from different matches of the query, in a unique way. Is the monotonicity a strict requirement for tractability? It turns out that the TC^0 upper bound can be strengthened towards all *existential queries*, i.e., existentially quantified formulas that allow negations in front of query atoms.

Theorem 5.13 MPD(∃FO) *is in* DLogTime*-uniform* TC^0 *in data complexity.*

Proof. Let $\mathcal{P}$ be a PDB, Q a ∃FO query and $p \in [0, 1)$, a threshold value. Analogously to the proof of Theorem 5.12, we can design an algorithm that initiates the database with the positive atoms from a match, while now also keeping the record of the negative atoms from this match. We can again enumerate all the matches, and for each such match, perform an iterated multiplication over the probabilities of the atoms. The only difference is that now we need to exclude the atoms from the database which appear negatively in the match to ensure the satisfaction of the query. Thus, for all probabilistic atoms $\langle a : q \rangle \in \mathcal{P}$, we check whether $\neg a$ appears in the match, and if so, then we multiply with $1-q$; otherwise, we check whether $q > 0.5$ and multiply with q, if this test is positive and multiply with $1 - q$, if this test is negative. It is then sufficient to compare the result of each multiplication with the threshold value p: if one of the resultant multiplications is greater than or equal to p, then the algorithm answers yes; otherwise, it answers no. It is easy to verify the correctness of this algorithm.

By similar arguments as in the proof of Theorem 5.12, we can check the matches in AC^0, perform the respective multiplications in TC^0 and the respective comparisons in AC^0. Thus, MPD(∃FO) is in DLogTime-uniform TC^0 in data complexity. □

In some sense, the tractability result presented implies that nonmonotonicity is not harmful if we restrict our attention to existential queries. This is because the nonmonotonicity involved here is of a limited type and does not lead to a combinatorial blow-up. This picture changes once we focus on universally quantified queries: nonmonotonicity combined with universal quantification creates nondeterministic choices, which we use to prove an NP-hardness result for the most probable database problem.

We note that this result is very similar to the hardness result obtained in (Gribkoff, Van den Broeck, and Suciu 2014a). Additionally, we show that the complexity bound is tight even if we consider FO queries.

Theorem 5.14 MPD(∀FO) *is* NP*-complete in data complexity, and so is* MPD(FO)*.*

Proof. We first show that MPD(FO) can be solved in NP. Let $\mathcal{P}$ be a PDB, Q a FO query and $p \in [0, 1)$, a threshold value. To solve the decision problem, we first guess

a world $\mathcal{D}$, and then verify both that the query is satisfied, i.e., $\mathcal{D} \models Q$, and that the threshold is met, i.e., $\mathrm{P}(\mathcal{D}) \geq p$. Note that all these computations can be done using a nondeterministic Turing machine since only the first step is nondeterministic, and both of the verification steps can be done in polynomial time in data complexity.

To prove the hardness, we provide a reduction from the satisfiability of propositional 3CNF formulas. Let $\varphi = \bigwedge_i \varphi_i$ be a propositional formula in 3CNF. Recall the ∀FO query Q_{SAT} (from Theorem 3.22), which we again use to encode the satisfaction conditions of φ. Without loss of generality, let us denote with $u_1, \ldots, u_n$ the propositional variables that appear in φ.

We then define the PDB $\mathcal{P}_\varphi$, depending on φ, as follows. For each propositional variable u_j, we add the probabilistic atom $\langle \mathsf{L}(u_j) : 0.5 \rangle$ to the PDB $\mathcal{P}_\varphi$. The clauses φ_j are described with the help of the predicates R_1, …, R_4, each of which corresponds to one type of clause (as in Theorem 3.24). The construction provided for Q_{SAT} and $\mathcal{P}_\varphi$ is clearly polynomial. Furthermore, the query is fixed, and only $\mathcal{P}_\varphi$ depends on φ. We now show that MPD can be used to answer the satisfiability problem of φ, using this construction.

Claim. The 3CNF formula φ is satisfiable if and only if there exists a database $\mathcal{D}$ induced by $\mathcal{P}_\varphi$ such that $\mathrm{P}(\mathcal{D}) \geq (0.5)^n$ and $\mathcal{D} \models Q_{\mathsf{SAT}}$ (where n is the number of variables appearing in φ).

To prove the claim, suppose that φ is satisfiable and let μ be such a satisfying assignment. We define a world $\mathcal{D}$ such that it contains all the atoms of the form $\mathsf{L}(u_j)$ if and only if $\mu(u_j) \mapsto 1$ in the given assignment. Moreover, $\mathcal{D}$ contains all the atoms which are assigned the probability 1 in $\mathcal{P}_\varphi$. It is easy to see that $\mathcal{D}$ is one of the worlds induced by $\mathcal{P}_\varphi$. Observe further that $\mathcal{P}_\varphi$ contains n nondeterministic atoms, each with 0.5 probability. By this argument, the probability of $\mathcal{D}$ is clearly $(0.5)^n$. It only remains to show that $\mathcal{D} \models Q_{\mathsf{SAT}}$, which is easy to verify.

For the other direction, let $\mathcal{D} \models Q_{\mathsf{SAT}}$ and $\mathrm{P}(\mathcal{D}) \geq (0.5)^n$ for some world $\mathcal{D}$. We define an assignment μ by setting the truth value of u_j to 1 if $\mathsf{L}(u_j) \in \mathcal{D}$, and to 0 otherwise. Every world contains exactly one assignment for every variable, by our construction. Thus, the assignment μ is well-defined. It is easy to verify that $\mu \models \varphi$. □

This concludes our analysis for the data complexity. To reflect the cases where the constraints themselves (specified in terms of queries) can be arbitrarily large, we next investigate the combined complexity for the most probable database problem.

Combined complexity results for MPD

As pointed before, once the query is considered to be part of the input, the complexity of the most probable database problem is typically dominated by the complexity of query answering in databases. Unsurprisingly, we obtain the following result.

Theorem 5.15 MPD(UCQ) *and* MPD(∃FO) *are both* NP*-complete in combined complexity.*

Proof. We first prove that MPD(∃FO) can be solved in NP. Let $\mathcal{P}$ be a PDB, Q a FO query and $p \in [0, 1)$, a threshold value. To solve the decision problem, we guess a world $\mathcal{D}$, and a mapping ϑ, which maps the variables in the query Q to the database

constants. Let us denote by $\vartheta(Q)$ the ground query that results from applying the mapping ϑ. Finally, we only need to verify whether $\vartheta(Q)$ is a match for the query, i.e., $\mathcal{D} \models \vartheta(Q)$ and whether the threshold is met by the database, i.e., $\mathrm{P}(\mathcal{D}) \geq p$. It is easy to see that the overall computation can be done using a single nondeterministic Turing machine as the verifications can be done in polynomial time. Thus, we conclude that $\mathsf{MPD}(\exists\mathsf{FO})$ is in NP, in combined complexity.

As for hardness, we provide a straightforward reduction from the UCQ evaluation problem in databases: given a UCQ Q and a database $\mathcal{D}$, decide whether $\mathcal{D} \models Q$. This problem is NP-hard in combined complexity (even if we assume that the arity of the predicates is bounded). To simulate this problem, we simply define a PDB $\mathcal{P}$ which contains all the atoms from $\mathcal{D}$ with probability 1. Then, we immediately obtain that $\mathcal{D} \models Q$ if and only if there is a database $\mathcal{D}'$ such that $\mathrm{P}(\mathcal{D}') \geq 1$ and $\mathcal{D}' \models Q$. □

In essence, this shows that MPD is not harder than query answering for UCQ and $\exists\mathsf{FO}$ queries in combined complexity. This holds since the nondeterministic computation for the most probable database, and for the mapping (to find a match) can be combined into a single nondeterministic machine. This approach clearly does not work for $\forall\mathsf{FO}$ queries, for which query satisfaction test is coNP-complete in combined complexity. Indeed, in what follows, we show that the computational complexity goes one level higher in the polynomial hierarchy for $\forall\mathsf{FO}$ queries.

Theorem 5.16 $\mathsf{MPD}(\forall\mathsf{FO})$ *is* Σ_2^{P}*-complete in combined complexity.*

Proof. Let $\mathcal{P}$ be a PDB, Q a $\forall\mathsf{FO}$ query and $p \in [0, 1)$, a threshold value. To prove membership, consider a nondeterministic Turing machine with a (co)NP oracle: given a PDB $\mathcal{P}$, a universally quantified query Q, and a threshold $p \in (0, 1]$, we can decide whether there exists a database $\mathcal{D}$ such that $\mathrm{P}(\mathcal{D}) \geq p$, by first guessing a database $\mathcal{D}$, and verifying whether (i) $\mathrm{P}(\mathcal{D}) \geq p$ and (ii) $\mathcal{D} \models Q$. Verification of (i) can be done in deterministic polynomial time and (ii) can be done in coNP, using the oracle. This proves a Σ_2^{P} upper bound.

To show hardness, we provide a reduction from the validity problem for quantified Boolean formulas of the form $\Phi = \exists u_1, \ldots, u_n \, \forall v_1, \ldots, v_m \, \varphi$, where φ is a propositional formula. This problem is Σ_2^{P}-complete (Stockmeyer 1976). We use the PDB $\mathcal{P}_\Phi$ that contains all atoms $\langle \mathsf{L}(u_j) : 0.5 \rangle$ for the existentially quantified variables u_j, and the two additional atoms $\langle \mathsf{L}(0) : 0 \rangle$ and $\langle \mathsf{L}(1) : 1 \rangle$. The query is then defined as $Q_\Phi = \forall v_1, \ldots, v_m \, \psi$, where ψ is obtained from φ by replacing all propositional variables u by $\mathsf{L}(u)$. Importantly, in this construction, existentially quantified variables in φ are viewed as constants in the query, and universally quantified variables in φ are still universally quantified in the query, but now range over the database constants instead of the truth values true and false. Obviously, this construction is polynomial in the size of φ; thus, it is sufficient to prove the following claim.

Claim. The QBF Φ is valid if and only if there exists a database $\mathcal{D}$ induced by $\mathcal{P}_\Phi$ that satisfies Q_Φ such that $\mathrm{P}(\mathcal{D}) \geq (0.5)^n$.

Assume that Φ is valid. Then, there exists a valuation μ for $u_1, \ldots, u_n$ such that all extensions τ to $v_1, \ldots, v_m$ satisfy φ. We choose the database $\mathcal{D}$ such that $\mathsf{L}(u_j) \in \mathcal{D}$ if and only if $\mu(u_j) = 1$, $\mathsf{L}(1) \in \mathcal{D}$, and $\mathsf{L}(0) \notin \mathcal{D}$. Hence, we obtain that $\mathrm{P}(\mathcal{D}) = (0.5)^n$. To

show that $\mathcal{D} \models Q_\Phi$, consider an assignment ι for $v_1, \ldots, v_m$ in Q_Φ that assigns a constant to each v_j. We define the extension τ of μ to the propositional variables $v_1, \ldots, v_m$ by setting $\tau(v_j) = 1$ if and only if $\mathsf{L}(\iota(v_j)) \in \mathcal{D}$. Since by assumption τ satisfies φ, it must be the case that ι satisfies ψ.

Conversely, let $\mathcal{D} \models Q_\varphi$ be such that $\mathrm{P}(\mathcal{D}) \geq (0.5)^n$. We define the valuation μ for $u_1, \ldots, u_n$ by setting $\mu(u_j) = 1$ iff $\mathsf{L}(u_j) \in \mathcal{D}$. Consider now any extension τ of μ to $v_1, \ldots, v_m$. We obtain a corresponding assignment for the variables in Q_Φ by setting $\iota(v_j) = \tau(v_j)$. Since $\mathcal{D}$ must satisfy $\mathsf{L}(1)$ and cannot satisfy $\mathsf{L}(0)$, we know from $\mathcal{D} \models Q_\Phi$ that τ satisfies φ. □

Our final result concerns first-order queries, for which we show that the PSPACE upper bound for query evaluation is preserved also for MPD.

Theorem 5.17 MPD(FO) *is* PSPACE*-complete in combined complexity for* FO.

Proof. Let $\mathcal{P}$ be a PDB, Q a FO query and $p \in [0, 1)$, a threshold value. As for membership, we can enumerate all exponentially many databases $\mathcal{D}$ induced by $\mathcal{P}$ in polynomial space, and for each of them, check whether $\mathcal{D} \models Q$ and whether $\mathrm{P}(\mathcal{D}) \geq p$, both of which can again be done in polynomial space.

Hardness is immediate since the query evaluation problem in databases for first-order queries is already PSPACE-hard in combined complexity (even if we assume that the arities of the predicates are bounded). To simulate this problem, we define a PDB $\mathcal{P}$ that contains all the atoms from a given arbitrary $\mathcal{D}$ with probability 1. Then, for any FO query Q, we have that $\mathcal{D} \models Q$ if and only if there is a database $\mathcal{D}'$ induced by $\mathcal{P}$ such that $\mathrm{P}(\mathcal{D}') \geq 1$ and $\mathcal{D}' \models Q$. □

This concludes our analysis for the most probable database problem in the context of PDBs, and this problem is revisited later for ontology-mediated queries. We now present an overview of the results.

Overview of the Complexity Results for MPD

All of the complexity results for the most probable database problem are summarized in Table 5.5. One of the major results asserts that MPD is tractable for *unions of conjunctive queries* in data complexity, which is then strengthened to the class of existential queries. In more concrete terms, our results imply that the most probable database for these query languages can be computed in logarithmic space and the overall computation can be encoded uniformly into a class of TC^0 circuits.

Surprisingly, we observe a sharp contrast between the complexity of MPD(∃FO) and MPD(∀FO): the problem for universal queries appears one level higher in the polynomial hierarchy in all cases. This contrast is sharper in data complexity under widely accepted complexity-theoretic assumptions: existential queries are highly tractable (in TC^0), whereas universal queries are NP-hard for MPD. This latter has already been observed in (Gribkoff, Van den Broeck, and Suciu 2014a); it is easy to show that MPD remains in NP for the class of first-order queries in data complexity.

In combined complexity, the complexity of MPD is typically dominated by the complexity of query evaluation in databases with the only exception being universal queries. That is, even though query evaluation is coNP-complete for ∀FO queries, MPD turns out

Table 5.5: Data, bounded-arity and combined complexity results for MPD for database queries. Results with '$\leq$' denote membership; all the rest are completeness results.

Query Languages	**Most Probable Database in PDBs**		
	data	bounded arity	combined
UCQ	$\leq$ TC^0 [Theorem 5.12]	NP [Theorem 5.15]	NP [Theorem 5.15]
$\exists$FO	$\leq$ TC^0 [Theorem 5.13]	NP [Theorem 5.15]	NP [Theorem 5.15]
$\forall$FO	NP [Gribkoff et al.]	Σ_2^{P} [Theorem 5.16]	Σ_2^{P} [Theorem 5.16]
FO	NP [Theorem 5.14]	PSpace [Theorem 5.17]	PSpace [Theorem 5.17]

to be one level higher in the polynomial hierarchy for these queries; namely, Σ_2^{P}-complete. Note also that all the combined complexity results presented here are in their strongest form in the following sense: all the hardness results already apply even if we assume that the arities of the predicates are bounded by a fixed constant.

The complexity bounds are tight for all cases except for the TC^0 upper bounds. Unfortunately, it remains *open* whether the given TC^0 bound is tight for both UCQ and $\exists$FO queries. At first sight, a lower bound may seem obvious as *i*) there is a multiplication involved in the computation of MPD and *ii*) even binary integer multiplication is known to be TC^0-hard under DLogTime-reduction (Immerman 1999). However, it is not clear how to simulate multiplication in our decision problem since the threshold (that relates to the result of the multiplication) is itself the input to the problem. In order to set the threshold appropriately, we need to know something about the result of the multiplication. This cyclic dependency indicates that we probably need a problem where the threshold can be set without any prior knowledge regarding the multiplication. We next define such a problem.

Definition 5.18 (MSB of multiplication) Given two integers l_1 and l_2, each with exactly with m bits, decide whether the *most significant bit (MSB) of the multiplication* $l_1 \cdot l_2$ is 1, provided that the result is printed in $2m$ bits with possibly leading zero's. $\Diamond$

Obviously, this is a special case of *binary multiplication*; that is, given integers l_1 and l_2, the problem of deciding whether the *i-th bit* of the multiplication is 1 (Hesse, Allender, and Barrington 2002). Note also that these problems generalize to iterated multiplication in the obvious way. The reduction from deciding the MSB of binary multiplication to our problem is straight-forward as we can simply set the threshold to (0.5). It is easy to see that our problem is at least as hard as deciding the MSB of binary multiplication.

To the best of our knowledge, deciding the MSB of binary multiplication has not been considered in the literature before. Thus, it is *open* whether deciding the MSB of binary multiplication is TC^0-hard. Unfortunately existing reductions for proving hardness of multiplication do not apply to the most significant bit; in particular, such reductions deliberately use the medium bits, which are somewhat more difficult to compute than the most significant bit. We explicitly introduce this problem as it may be of independent interest.

Table 5.6: The most probable hypothesis for the query Q_{vf} over $\mathcal{P}_v$.

Vegetarian	
alice	0.7
bob	0.9
chris	0.6

FriendOf		
alice	bob	0.7
alice	chris	0.8
bob	chris	0.1

Eats		
bob	spinach	0.7
chris	mussels	0.8
alice	broccoli	0.2

Meat	
shrimp	0.7
mussels	0.9
seahorse	0.3

5.2.2 The Most Probable Hypothesis Problem

Finding the most probable database is important, as it identifies the most likely state of a probabilistic database relative to a query. However, it has certain limitations, which are analogous to the limitations of finding the *most probable explanations* in PGMs (Koller and Friedman 2009). Most importantly, one is always forced to choose a *complete* database although the query usually affects only a subset of the atoms. In other words, it is usually not the case that the whole database is responsible for the goal query to be satisfied. To be able to more precisely pinpoint the explanations of a query, we introduce the following more intricate notion.

Definition 5.19 The *most probable hypothesis* for a query Q over a PDB $\mathcal{P}$ is

$$\arg\max_{\mathcal{H} \models Q} \sum_{\mathcal{D} \models \mathcal{H}} \mathrm{P}(\mathcal{D}),$$

where $\mathcal{H}$ ranges over sets of atoms t and negated atoms $\neg t$ such that t occurs in $\mathcal{P}$, and $\mathcal{H} \models Q$ holds if and only if all worlds induced by $\mathcal{P}$ that satisfy $\mathcal{H}$ also satisfy Q. $\Diamond$

Intuitively, the most probable hypothesis contains atoms only if they contribute to the satisfaction of the query. In other words, for the most probable hypothesis, we allow partial explanations. It is still the case that an explanation has to satisfy the query, but to do so, it does not need to make a decision for all of the database atoms. Indeed, it is possible to specify some positive and negative atoms that ensure the satisfaction of the query, regardless of the truth value of the remaining database atoms.

Conversely, any database $\mathcal{D}$ with $\mathcal{D} \models \mathcal{H}$ must satisfy Q, and thus the most probable database can be seen as a special case of the most probable hypothesis that has to contain all atoms from a PDB (positively or negatively). We denote the sum inside the maximization, i.e., the probability of the explanation by $\mathrm{P}(\mathcal{H})$. Differently from PGMs, the probability of the explanation can be computed by simply taking the product of the probabilities of the (negated) atoms in $\mathcal{H}$. This is a consequence of the independence assumption and will influence the complexity results as we will elaborate later.

Example 5.20 Consider again our running example with the PDB $\mathcal{P}_v$ and recall the query $Q_{\mathsf{vf}} \coloneqq Q_{\mathsf{veg}} \wedge Q_{\mathsf{fr}}[x/\mathsf{bob}]$ where

$$Q_{\mathsf{veg}} \coloneqq \forall x, y \ \neg\mathsf{Vegetarian}(x) \vee \neg\mathsf{Eats}(x, y) \vee \neg\mathsf{Meat}(y),$$
$$Q_{\mathsf{fr}}[x/\mathsf{bob}] \coloneqq \exists y \ Q_{\mathsf{veg}}(\mathsf{bob}) \wedge \mathsf{FriendOf}(\mathsf{bob}, y) \wedge Q_{\mathsf{veg}}(y).$$

Recall that the most probable database for Q_{vf} contains many redundant atoms. The most probable hypothesis $\mathcal{H}$ for the query Q_{vf} contains only 4 atoms as given in Table 5.6:

as before orange highlighting denotes positive atoms, while the red one denotes negated atoms, i.e., $\neg\mathsf{Eats}(\mathsf{chris}, \mathsf{mussels})$. Notice that all of these atoms directly influence the satisfaction of the query and thus are part of the explanation. Since the most probable hypothesis contains less atoms, in general, it is more informative than the most probable database. Moreover, we can compute the probability of the hypothesis by $\mathrm{P}(\mathcal{H}) = 0.9 \cdot 0.6 \cdot 0.1 \cdot (1 - 0.8) = 0.0108$. ◊

Whereas the most probable database represents full knowledge about all facts, which corresponds to the common closed-world assumption for (probabilistic) databases, the most probable hypothesis may leave atoms of $\mathcal{P}$ unresolved, which can be seen as a kind of open-world assumption (although the atoms that do not occur in $\mathcal{P}$ are still false). MPH is defined as a decision problem as follows.

Definition 5.21 (MPH) Let Q be a query, $\mathcal{P}$ a PDB, and $p \in (0, 1]$ a threshold. MPH is the problem of deciding whether there exists a hypothesis $\mathcal{H}$ that satisfies Q with $\mathrm{P}(\mathcal{H}) \geq p$. MPH is parametrized with a particular query language; thus, we write MPH(Q) to define MPH on the class Q of queries. ◊

Data complexity results for MPH

We again start our analysis with *unions of conjunctive queries* and show that, as is the case for MPD, MPH can also be solved using at most *logarithmic space* and *polynomial time*.

Theorem 5.22 MPH(UCQ) *is in* DLogTime*-uniform* TC^0 *in data complexity.*

Proof. Let $\mathcal{P}$ be a PDB, Q a UCQ and $p \in [0, 1)$, a threshold value. It has already been observed that the satisfaction relation for union of conjunctive queries is monotone, i.e., the fact that once a database satisfies a UCQ, then any superset of this database satisfies the UCQ. Clearly, this also applies to partial explanations: the databases extending the hypothesis $\mathcal{H}$ satisfy the query only if $\mathcal{H}$ (extended with all atoms that have probability 1) is already a match for the query. This means that the hypothesis must be a subset of a ground instance of one of the disjuncts of the UCQ Q. The major difference from MPD is that, once the explanation is found, we do not need to consider the other database atoms in the PDB. As before, there are only polynomially many such hypotheses in the data complexity, and they can be encoded uniformly into AC^0 circuits, as in the proof of Theorem 5.12. Moreover, their probabilities can be computed in TC^0 and the comparison with the threshold p can be done again in AC^0. This puts MPH(UCQ) in DLogTime-uniform TC^0 in data complexity. □

Recall that, for MPD, it was possible to generalize the data tractability result from unions of conjunctive queries to existential queries. It is therefore interesting to know whether the same holds for MPH. Does MPH remain tractable if we consider existential queries? The answer is unfortunately negative: to be able to verify a test such as $\mathcal{H} \models Q$, we need to make sure that all extensions $\mathcal{D}$ of $\mathcal{H}$ satisfy the query and this test is hard once we allow negations in front of query atoms. We illustrate the effect of negations with a simple example.

Example 5.23 Consider the following ∃FO query

$$Q := \exists x, y\ (\mathsf{A}(x) \wedge \neg\mathsf{B}(x,y)) \vee (\neg\mathsf{A}(x) \wedge \neg\mathsf{B}(x,y)).$$

To decide MPH on an arbitrary PDB, we could walk through all (partial) matches for the query and then verify $\mathcal{H} \models Q$. However, observe that this verification is not in polynomial time, in general, as there are interactions between the query atoms and these interactions need to be captured by the explanation itself. For instance, the A-atoms in Q are actually redundant and it is enough to find an explanation that satisfies $\exists x, y\ \neg\mathsf{B}(x,y)$ for some mapping. ◊

We next provide a reduction from the validity problem of 3DNF formulas, proving that MPH is coNP-hard for existential queries. It is easy to see that this is also a matching upper bound.

Theorem 5.24 MPH(∃FO) *is* coNP*-complete in data complexity.*

Proof. Let $\mathcal{P}$ be a PDB, Q an ∃FO query and $p \in [0,1)$ a threshold value. As for membership, consider a nondeterministic Turing machine, which enumerates all partial matches forming the hypothesis $\mathcal{H}$ and answers yes if and only if $\mathrm{P}(\mathcal{H}) \geq p$ and there is *no* database $\mathcal{D}$ that satisfies $\mathcal{D} \models \mathcal{H}$ while $\mathcal{D} \not\models Q$.

To prove the hardness, we provide a reduction from the validity of propositional 3DNF formulas. Let $\varphi = \bigvee_i \varphi_i$ be a propositional formula in 3DNF. We first define the following ∃FO query

$$\begin{aligned} Q_{\mathsf{VAL}} := \exists x, y, z\ (\ \ \mathsf{L}(x) \wedge\ \ \mathsf{L}(y) \wedge\ \ \mathsf{L}(z) \wedge \mathsf{R}_1(x,y,z)) \vee \\ (\neg\mathsf{L}(x) \wedge\ \ \mathsf{L}(y) \wedge\ \ \mathsf{L}(z) \wedge \mathsf{R}_2(x,y,z)) \vee \\ (\neg\mathsf{L}(x) \wedge \neg\mathsf{L}(y) \wedge\ \ \mathsf{L}(z) \wedge \mathsf{R}_3(x,y,z)) \vee \\ (\neg\mathsf{L}(x) \wedge \neg\mathsf{L}(y) \wedge \neg\mathsf{L}(z) \wedge \mathsf{R}_4(x,y,z)) \end{aligned}$$

which is later used to encode the validity conditions of φ. Without loss of generality, let us denote with $u_1, \ldots, u_n$ the propositional variables that appear in φ. We then define the PDB $\mathcal{P}_\varphi$, depending on φ, as follows.

- For each propositional variable u_j, we add the probabilistic atom $\langle \mathsf{L}(u_j) : 0.5 \rangle$ to the PDB $\mathcal{P}_\varphi$.
- The conjuncts φ_j are described with the help of the predicates R_1, …, R_4, each of which corresponds to one type of conjunct. For example, if we have a clause $\varphi_j = u_1 \wedge \neg u_2 \wedge \neg u_4$, we add the atom $\langle \mathsf{R}_3(u_4, u_2, u_1) : 1 \rangle$ to $\mathcal{P}_\Phi$, which enforces via Q_{VAL} that the conjunct that includes all atoms $\neg\mathsf{L}(u_4)$, $\neg\mathsf{L}(u_2)$ and $\mathsf{L}(u_1)$ can be true. All other atoms $\mathsf{R}_i(u_k, u_l, u_m)$ that do not correspond in such a way to one of the clauses, we add with probability 0 to $\mathcal{P}_\Phi$.

The construction provided for Q_{VAL} and $\mathcal{P}_\varphi$ is clearly polynomial. Furthermore, the query is fixed, and only $\mathcal{P}_\varphi$ depends on φ. We now show that MPH can be used to answer the validity problem of φ, using this construction.

Claim. The 3DNF formula φ is valid if and only if there exists a hypothesis $\mathcal{H}$ over $\mathcal{P}_\Phi$ such that $\mathrm{P}(\mathcal{H}) \geq 1$ and $\mathcal{H} \models Q_{\mathsf{VAL}}$.

To prove the claim, suppose that φ is valid. We show that the empty hypothesis $\mathcal{H} = \emptyset$ satisfies both $\mathrm{P}(\mathcal{H}) \geq 1$ and $\mathcal{H} \models Q_{\mathsf{VAL}}$. It is easy to see that $\mathrm{P}(\mathcal{H}) = 1$ in $\mathcal{P}_\varphi$ as it encodes all possible databases, i.e., worlds. Let us assume by contradiction that there exists a database $\mathcal{D}$ that extends the hypothesis, $\mathcal{H} \models \mathcal{D}$, but does not satisfy the query, i.e., $\mathcal{D} \not\models Q_{\mathsf{VAL}}$. Then, we can use this database to define a valuation for the given propositional formula: define an assignment μ by setting the truth value of u_j to 1 if $\mathsf{L}(u_j) \in \mathcal{D}$, and to 0; otherwise. Every world contains exactly one assignment for every variable, by our construction. Thus, the assignment μ is well-defined. But then, it is easy to see that this implies $\mu \not\models \varphi$, which contradicts the validity of φ.

For the other direction, let $\mathcal{H}$ be a hypothesis such that $\mathrm{P}(\mathcal{H}) \geq 1$ and $\mathcal{H} \models Q_{\mathsf{VAL}}$. This can only be the case if the $\mathcal{H}$ is the empty set (as otherwise $\mathrm{P}(\mathcal{H}) \leq 0.5$). This means that any database $\mathcal{D}$ induced by $\mathcal{P}_\varphi$ must satisfy the query. It is easy to see that every database is in one-to-one correspondence with a propositional assignment; thus, we conclude the validity of Q_{VAL}. □

Having shown that MPH is harder than MPD for existential queries, one may wonder whether this is also the case for universal queries. We now show that MPH has the same complexity as MPD for universal queries.

Theorem 5.25 MPH(∀FO) *is* NP*-complete in data complexity.*

Proof. Let $\mathcal{P}$ be a PDB, Q a ∀FO query and $p \in [0, 1)$, a threshold value. To show membership, we nondeterministically guess a hypothesis $\mathcal{H}$ such that the satisfaction relation $\mathcal{H} \models Q$ is ensured. To do so, assume without loss of generality that the universal query is of the form

$$Q \coloneqq \forall x_1, \ldots, x_n \bigwedge_i q_i(x_1, \ldots, x_n),$$

for some finite number $i, n > 0$, where each $q_i(x_1, \ldots, x_n)$ is a clause over $x_1, \ldots, x_n$. Our goal is to find a hypothesis that satisfies this query and that meets the threshold value p. To satisfy a universal query, we need to identify all database atoms that could possibly invalidate the query and rule them out. Thus, we consider the negation of the given query

$$\neg Q = \exists x_1, \ldots, x_n \bigvee_i \neg q_i(x_1, \ldots, x_n),$$

which encodes all database atoms that could possibly invalidate the original query Q. Furthermore, to make the effect of the database atoms more concrete, we consider all possible groundings of this query. Let us denote by $\neg Q[x_1/a_1, \ldots x_n/a_n]$ a grounding with database constants a_i. Clearly, there are polynomially many such groundings in data complexity. We need to ensure that none of the clauses in any of the groundings is satisfied by the hypothesis. Note that this can be achieved by including, for every clause, an atom into the hypothesis that contradicts the respective clause. For example, suppose that a grounding of the query is

$$(\mathsf{A}(a) \wedge \neg \mathsf{B}(b)) \vee \ldots \vee (\neg \mathsf{C}(c) \wedge \neg \mathsf{D}(d)).$$

Then, for the first clause, we have to either add the atom $\neg \mathsf{A}(a)$ or the atom $\mathsf{B}(b)$ to the hypothesis, and similarly for the last clause. The subtlety is that we cannot deterministically decide which atoms to include into the hypothesis (as there are interactions across

clauses as well as across different groundings). Therefore, we nondeterministically guess each such choice. In essence, while constructing the hypothesis, we rule out everything that could invalidate the original query. As a consequence, we ensure that $\mathcal{H} \models Q$. It only remains to check whether $\mathrm{P}(\mathcal{H}) \geq p$, which can be done in polynomial time. Thus, we obtain an NP upper bound in data complexity.

Hardness is analogous to the proof of Theorem 5.14. Therefore, we consider the same construction; namely, given a 3CNF formula φ, we define Q_{SAT} and $\mathcal{P}_\varphi$, as in Theorem 5.14 where only $\mathcal{P}_\varphi$ depends on φ. The construction is polynomial; thus, it suffices to prove the correctness of the following claim.

Claim. The 3CNF formula φ is satisfiable if and only if there exists a hypothesis $\mathcal{H}$ over $\mathcal{P}_\varphi$ such that $\mathrm{P}(\mathcal{H}) \geq (0.5)^n$ and $\mathcal{H} \models Q_{\mathsf{SAT}}$ (where n is the number of variables appearing in φ).

Soundness of this claim is immediate as we can define a hypothesis $\mathcal{H}$ based on a satisfying assignment μ of φ, as in Theorem 5.14, whereby we ensure that $\mathcal{H} \models Q_{\mathsf{SAT}}$ and $\mathrm{P}(\mathcal{H}) \geq (0.5)^n$. For completeness, it is sufficient to observe that whenever there exists a hypothesis $\mathcal{H}$ with $\mathcal{H} \models Q_{\mathsf{SAT}}$ and $\mathrm{P}(\mathcal{H}) \geq (0.5)^n$, there is at least one (although possibly more) database $\mathcal{D}$ such that $\mathcal{H} \models \mathcal{D}$ and $\mathcal{D} \models Q_{\mathsf{SAT}}$, which we can use to define a satisfying assignment μ of φ. Thus, we conclude that MPH(∀FO) is NP-complete in data complexity. □

MPH is coNP-complete for ∃FO queries (by Theorem 5.24) and NP-complete for ∀FO queries (by Theorem 5.25). By considering a particular query, which combines the power of existential and universal queries, we are able to show Σ_2^{P}-hardness for MPH.

Theorem 5.26 MPH(FO) *is* Σ_2^{P}*-complete in data complexity.*

Proof. Let $\mathcal{P}$ be a PDB, Q a FO query and $p \in [0, 1)$, a threshold value. Consider a nondeterministic Turing machine with a (co)NP oracle: given a PDB $\mathcal{P}$, a first-order query Q, and a threshold $p \in (0, 1]$, we can decide whether there exists a hypothesis $\mathcal{H}$ such that $\mathrm{P}(\mathcal{H}) \geq p$ by first guessing a hypothesis $\mathcal{H}$, and verifying whether (i) $\mathrm{P}(\mathcal{H}) \geq p$ and (ii) for *all* databases $\mathcal{D}$ that extend $\mathcal{H}$ and are induced by $\mathcal{P}$, it holds that $\mathcal{D} \models Q$. Verification of (i) can be done in deterministic polynomial time and (ii) can be done in coNP (the complement is equivalent to the existence of an extension $\mathcal{D}$ and a valuation for the query variables that falsifies Q). This shows that MPH(FO) is in Σ_2^{P} in data complexity.

As for hardness, we provide a reduction from validity of quantified Boolean formulas of the form $\Phi = \exists u_1, \ldots, u_n \forall v_1, \ldots, v_m\, \varphi$, where φ is in 3DNF. Checking validity of such formulas is known to be Σ_2^{P}-complete. For the reduction, we consider the following query $Q = Q_{\mathsf{VAL}} \wedge Q_s$ where Q_{VAL} is the ∃FO query from Theorem 5.24 that encodes the validity conditions and Q_s is a ∀FO query defined as

$$Q_s := \forall x(\neg \mathsf{E}(x) \wedge \mathsf{L}(x)) \vee (\mathsf{E}(x) \wedge \neg \mathsf{L}(x)) \vee \mathsf{F}(x).$$

Moreover, we define a PDB $\mathcal{P}_\Phi$ such that

- for each variable u that appears in φ, $\mathcal{P}_\Phi$ contains the atom $\langle \mathsf{L}(u) : 0.5 \rangle$.
- for every existentially quantified variable u_j, $\mathcal{P}_\Phi$ contains the atom $\langle \mathsf{E}(u_j) : 0.5 \rangle$.

- for every universally quantified variable v_j, $\mathcal{P}_\Phi$ contains the atom $\langle \mathsf{F}(v_j) : 1 \rangle$.
- every conjunction in φ are described with the help of the predicates R_1, …, R_4, each of which corresponds to one type of conjunctive clause. For example, if we have $\varphi_j = u_1 \wedge \neg u_2 \wedge \neg u_4$, we add the atom $\langle \mathsf{R}_3(u_4, u_2, u_1) : 1 \rangle$ to $\mathcal{P}_\Phi$, which enforces via Q_{VAL} that the clause that includes all atoms $\neg\mathsf{L}(u_4)$, $\neg\mathsf{L}(u_2)$ and $\mathsf{L}(u_1)$ can be true. Moreover, all the remaining R_i-atoms have probability 0.

In this construction, Q_{VAL} encodes the 3DNF and Q_s helps us to distinguish between the existentially and universally quantified variables through the E- and F-atoms.

Claim. The quantified Boolean formula Φ is valid if and only if there exists a hypothesis $\mathcal{H}$ over $\mathcal{P}_\Phi$ such that $\mathrm{P}(\mathcal{H}) \geq (0.5)^{2n}$ and $\mathcal{H} \models Q$.

Suppose that Φ is valid. Then, there exists a valuation μ of $u_1, \ldots, u_n$, such that all valuations τ that extend this partial valuation (by assigning truth values to $v_1, \ldots, v_m$) satisfy φ. We define a hypothesis $\mathcal{H}$ depending on μ as follows. For all assignments $u_j \mapsto 1$ in μ, we add $\mathsf{L}(u_j)$ to $\mathcal{H}$; if, on the other hand, $u_j \mapsto 0$ in μ we add $\neg\mathsf{L}(u_j)$ to $\mathcal{H}$. Moreover, to satisfy the query Q_s, for every $\mathsf{L}(u_j) \in \mathcal{H}$, we add $\neg\mathsf{E}(u_j)$ to $\mathcal{H}$, and analogously, for every $\neg\mathsf{L}(u_j) \in \mathcal{H}$, we add $\mathsf{E}(u_j)$ to $\mathcal{H}$. By this construction, there are clearly $2n$ atoms in $\mathcal{H}$, each of which has the probability 0.5 in $\mathcal{P}_\varphi$. Hence, it holds that $\mathrm{P}(\mathcal{H}) = (0.5)^{2n}$. Finally, it is sufficient to observe that all databases $\mathcal{D}$ which extend $\mathcal{H}$ must satisfy the query Q as every such database is in one-to-one correspondence with a valuation τ that extends μ.

For the other direction, we assume that there exists a hypothesis $\mathcal{H}$ over $\mathcal{P}_\Phi$ such that $\mathrm{P}(\mathcal{H}) \geq (0.5)^{2n}$ and $\mathcal{H} \models Q$. This implies that $\mathcal{H}$ contains at most $2n$ atoms that have probability 0.5 in $\mathcal{P}_\varphi$ (and possibly some deterministic atoms). Furthermore, since $\mathcal{H} \models Q_s$, we know that $\mathcal{H}$ contains each E-atom either positively or negatively, and it also contains the complementary L-atom. Since these are already $2n$ atoms, $\mathcal{H}$ cannot contain any L-atoms for the universally quantified variables v_j. We can thus define a valuation μ for $u_1, \ldots, u_n$ simply by setting $u_j \mapsto 1$ if $\mathsf{L}(u_j) \in \mathcal{H}$ and $u_j \mapsto 0$ if $\neg\mathsf{L}(u_j) \in \mathcal{H}$. It is easy to see that the extensions τ of μ are in one-to-one correspondence with the databases that extend $\mathcal{H}$, and that φ evaluates to true for all of these assignments. □

With this result, we conclude the data complexity analysis for MPH and investigate the combined complexity of MPH next.

Combined complexity results for MPH

In the combined complexity, the first result is a rather trivial one, and shows that the complexity of UCQ evaluation is a dominating factor for MPH(UCQ).

Theorem 5.27 MPH(UCQ) *is* NP*-complete in combined complexity.*

Proof. Let $\mathcal{P}$ be a PDB, Q a UCQ and $p \in [0, 1)$, a threshold value. As for the combined complexity, we can guess a hypothesis, compare its probability to the threshold in polynomial time, and verify that it satisfies one of the disjuncts of Q. MPH(UCQ) is therefore in NP in combined complexity.

It is easy to adopt the naïve reduction from the proof of Theorem 5.15 in order to show hardness. More precisely, given a database $\mathcal{D}$ and a query Q, we define a PDB

$\mathcal{P}$, which contains all the atoms from $\mathcal{D}$ with probability 1. We obtain that $\mathcal{D} \models Q$ if and only if there is a hypothesis $\mathcal{H}$ such that $\mathrm{P}(\mathcal{H}) \geq 1$ and $\mathcal{H} \models Q$. This gives us a matching lower bound and we conclude that $\mathsf{MPH}(\mathsf{UCQ})$ is NP-complete in combined complexity. □

Our next result is a surprising one, as it shows that the complexity of MPH for $\exists\mathsf{FO}$ queries is two levels higher in the polynomial hierarchy than the complexity of MPH for unions of conjunctive queries. This is also in contrast with $\mathsf{MPD}(\exists\mathsf{FO})$ that is in NP in combined complexity.

Theorem 5.28 $\mathsf{MPH}(\exists\mathsf{FO})$ *is* Σ_3^{P}*-complete in combined complexity.*

Proof. Let $\mathcal{P}$ be a PDB, Q an $\exists\mathsf{FO}$ and $p \in [0, 1)$, a threshold value. For membership, consider a nondeterministic Turing machine with a Σ_2^{P} oracle. We can first guess a hypothesis $\mathcal{H}$, and verify first (in polynomial time) that the threshold is met, i.e., $\mathrm{P}(\mathcal{H}) \geq p$. It only remains to verify that for all databases $\mathcal{D}$ induced by the PDB extending $\mathcal{H}$, it holds that $\mathcal{D} \models Q$. Observe that the complement of the latter problem can be verified in Σ_2^{P}: guess a database $\mathcal{D}$ such that $\mathcal{D}$ extends $\mathcal{H}$, and check in coNP whether $\mathcal{D} \not\models Q$. This yields a Σ_3^{P} upper bound for the problem.

We now show Σ_3^{P}-hardness. Consider a quantified Boolean formula of the form

$$\Phi = \exists u_1, \ldots, u_n \forall v_1, \ldots, v_m \exists w_1, \ldots, w_k \, \varphi,$$

where $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_l$ is in CNF. Checking validity of such formulas is known to be Σ_3^{P}-complete. We provide a reduction from this problem. We start by defining a PDB $\mathcal{P}_\Phi$ as follows:

- For every existentially quantified variable u_j, $\mathcal{P}_\Phi$ contains the atoms $\langle \mathsf{L}(u_j, 0) : 0.5 \rangle$ and $\langle \mathsf{L}(u_j, 1) : 0.5 \rangle$.
- For every universally quantified variable v_j, $\mathcal{P}_\Phi$ contains the atom $\langle \mathsf{S}(v_j) : 0.5 \rangle$.
- For the existentially quantified variables w_j, we only add the two atoms $\langle \mathsf{T}(1) : 1 \rangle$ and $\langle \mathsf{T}(0) : 0 \rangle$ to $\mathcal{P}_\Phi$.

We now construct a query from the given propositional formula:

- For every clause φ_j, we construct a disjunction ψ_j by replacing the propositional variables with appropriate L-atoms, S-atoms, or T-atoms. For instance, for $\varphi_j = u_1 \vee \neg u_2 \vee v_3 \vee \neg w_6$, we use

  $$\psi_j = \mathsf{L}(u_1, 1) \vee \mathsf{L}(u_2, 0) \vee \mathsf{S}(v_3) \vee \neg \mathsf{T}(w_6).$$

  The atom $\mathsf{L}(u_1, 1)$ indicates that the existentially quantified variable u_1 should be true, and dually for $\mathsf{L}(u_2, 0)$. The atom $\mathsf{S}(v_3)$ says that the universally quantified variable v_3 should be true, and we negate such atoms if the variable is negated in the clause. Finally, $\neg \mathsf{T}(w_6)$ expresses a similar condition on the remaining existentially quantified variable. Note that here w_6 is a variable, while u_1, u_2, v_3 are constants from $\mathcal{P}_\Phi$.
- For all existentially quantified variables u_i, we additionally define the formulas

  $$\psi_{e_i} = (\neg \mathsf{L}(u_i, 0) \vee \neg \mathsf{L}(u_i, 1)) \wedge (\mathsf{L}(u_i, 0) \vee \mathsf{L}(u_i, 1)).$$

– We finally construct the existential query

$$Q_\Phi = \exists w_1, \ldots, w_n \bigvee_{1 \le j \le l} \psi_j \wedge \bigwedge_{1 \le i \le n} \psi_{e_i}.$$

Intuitively, $\mathcal{P}_\Phi$ together with $\psi_1, \ldots, \psi_l$ encodes the satisfaction condition of the formula, while $\psi_{e_1}, \ldots, \psi_{e_n}$ force the hypothesis to contain the L-atoms for the existentially quantified variables $u_1, \ldots, u_n$.

Claim. The formula Φ is valid if and only if there exists a hypothesis $\mathcal{H}$ over $\mathcal{P}_\Phi$ such that $\mathrm{P}(\mathcal{H}) \ge (0.5)^{2n}$ and $\mathcal{H} \models Q_\Phi$.

Assume that Φ is valid. Then there exists a valuation μ for $u_1, \ldots, u_n$, such that all extentions τ of ν to $v_1, \ldots, v_m$ admit an extension ι to $w_1, \ldots, w_k$ that satisfies φ. We choose the hypothesis $\mathcal{H}$ as follows. We add $\mathsf{L}(u_j, 1)$ and $\neg\mathsf{L}(u_j, 0)$ to $\mathcal{H}$ if $\nu(u_j) = 1$, and otherwise we add $\neg\mathsf{L}(u_j, 1)$ and $\mathsf{L}(u_j, 0)$ to $\mathcal{H}$. Hence, we have $\mathrm{P}(\mathcal{H}) = (0.5)^{2n}$. Now, for any database $\mathcal{D}$ that is induced by $\mathcal{P}_\Phi$ and extends $\mathcal{H}$, we must have $\mathsf{T}(1) \in \mathcal{D}$ and $\mathsf{T}(0) \notin \mathcal{D}$, and for each universally quantified variable v_j, $\mathcal{D}$ fixes a truth value via the atom $\mathsf{S}(v_j)$. This defines an extension $\tau_\mathcal{D}$ of μ by setting $\tau_\mathcal{D}(v_j) = 1$ if and only if $\mathsf{S}(v_j) \in \mathcal{D}$. By assumption, we know that there is an extension $\iota_\mathcal{D}$ of $\tau_\mathcal{D}$ to the remaining variables such that φ is satisfied. We can hence satisfy Q_Φ in $\mathcal{D}$ by mapping each w_j to 1 if $\iota_\mathcal{D}(w_j) = 1$, and to 0 otherwise. This shows that $\mathcal{H} \models Q_\Phi$.

Conversely, suppose that $\mathrm{P}(\mathcal{H}) \ge (0.5)^{2n}$ for some hypothesis $\mathcal{H}$, i.e., $\mathcal{H} \models Q_\Phi$. By construction of the query, for every existentially quantified variable u_j, $\mathcal{H}$ must contain the two atoms $\mathsf{L}(u_j, 0)$ and $\mathsf{L}(u_j, 1)$, one positively and the other negatively, to satisfy the query. This implies that, for $\mathcal{H}$ to achieve the threshold $(0.5)^{2n}$, it must contain exactly two L-atoms for each existentially quantified variable u_j (and it can contain some deterministic atoms). To show that Φ is valid, we now define the partial assignment μ such that $\mu(u_j) = 1$ if $\mathsf{L}(u_j, 1) \in \mathcal{H}$ (thus, $\neg\mathsf{L}(u_j, 0) \in \mathcal{H}$), and $\mu(u_j) = 0$; otherwise. Consider now any extension τ of μ to $v_1, \ldots, v_m$, and construct the extension $\mathcal{D}$ of $\mathcal{H}$ by adding $\mathsf{S}(v_j)$ if and only if $\tau(v_j) = 1$. We must also add $\mathsf{T}(1)$ to $\mathcal{D}$. It is easy to see that $\mathrm{P}(\mathcal{D}) > 0$. Hence, by assumption there must be an instantiation η of the query variables $w_1, \ldots, w_k$ that satisfies Q_Φ. We define the extension ι_η of τ to the propositional variables $w_1, \ldots, w_k$ by setting $\iota_\eta(w_j) = 1$ iff $\mathsf{T}(\eta(w_j)) \in \mathcal{D}$. Due to the construction of Q_Φ, this extension must satisfy φ, and hence we know that Φ is valid.□

What appears to be also surprising is that MPH for universal queries is not as hard as MPH for existential queries.

Theorem 5.29 $\mathsf{MPH}(\forall\mathsf{FO})$ *is* Σ_2^P*-complete in combined complexity.*

Proof. Consider a nondeterministic Turing machine M with a (co)NP oracle: Given a PDB $\mathcal{P}$, a universal query Q, and a threshold $p \in (0, 1]$, we can decide whether there exists a hypothesis $\mathcal{H}$ such that $\mathrm{P}(\mathcal{H}) \ge p$ by first guessing a hypothesis $\mathcal{H}$, and verifying whether (i) $\mathrm{P}(\mathcal{H}) \ge p$ and (ii) for *all* databases $\mathcal{D}$ that extend $\mathcal{H}$ and are induced by $\mathcal{P}$, it holds that $\mathcal{D} \models Q$. Verification of (i) can be done in deterministic polynomial time and (ii) can be done in coNP (the complement is equivalent to the existence of an extension $\mathcal{D}$ and a valuation for the query variables that falsifies Q).

Table 5.7: Data, bounded-arity combined, combined complexity results for MPH for database queries. Result with '$\leq$' denotes membership; all the rest are completeness results.

Query Languages	Most Probable Hypothesis in PDBs		
	data	bounded-arity	combined
UCQ	$\leq \mathrm{TC}^0$ [Theorem 5.22]	NP [Theorem 5.27]	NP [Theorem 5.27]
∃FO	coNP [Theorem 5.24]	Σ_3^{P} [Theorem 5.28]	Σ_3^{P} [Theorem 5.28]
∀FO	NP [Theorem 5.25]	Σ_2^{P} [Theorem 5.29]	Σ_2^{P} [Theorem 5.16]
FO	Σ_2^{P} [Theorem 5.26]	PSpace [Theorem 5.30]	PSpace [Theorem 5.30]

To show hardness, we provide a reduction from the validity problem for quantified Boolean formulas of the form $\Phi = \exists u_1, \ldots, u_n \, \forall v_1, \ldots, v_m \, \varphi$, as in the proof of Theorem 5.16. The construction for the PDB $\mathcal{P}_\Phi$ and the query Q_Φ is exactly the same. The correctness of the following claim can be shown using analogous arguments to those in Theorem 5.16, we can show that the QBF Φ is valid if and only if there exists a hypothesis $\mathcal{H}$ over $\mathcal{P}_\Phi$ that satisfies Q_Φ such that $\mathrm{P}(\mathcal{H}) \geq (0.5)^n$. □

Our final result for this section concerns the combined complexity of MPH for FO queries. By similar arguments to those in Theorem 5.17, we immediately obtain a tight complexity bound for FO queries.

Theorem 5.30 MPH(FO) *is* PSpace*-complete in combined complexity.*

Overview of the Complexity Results for MPH

All of the results regarding the most probable hypothesis are summarized in Table 5.7. We first showed a data tractability result concerning unions of conjunctive queries analogous to MPD. As before, this is the only result with no matching lower bound. Unlike MPD, however, ∃FO queries are proven to be coNP-complete for MPH in data complexity. Besides, MPH remains NP-complete for ∀FO queries, but turns out to be Σ_2^{P}-complete for first-order queries in data complexity.

For the combined case, the complexity of MPH is typically higher than for MPD. This is an expected phenomenon. What appears to be more surprising at first sight is the following. Existential queries are more difficult than universal queries for MPH; that is, the opposite of what has been observed for MPD. Detailed insights on this observation are given in the respective proofs.

5.3 Related Work and Outlook

Diagnostic reasoning, also known as abductive reasoning (Eiter and Gottlob 1995) or explanation, is an important problem in artificial intelligence and its applications, such as natural language understanding, medical diagnosis, common-sense explanation, and pattern recognition.

Inspired by maximal posterior computations in PGMs, such as MAP and MPE, we studied two decision problems for probabilistic databases, namely MPD and MPH. Note that there exist other variants of MAP and MPE (Kwisthout 2011), and there are different naming conventions across communities; here, we use the terminology from the BN literature (Darwiche 2009).

As pointed out before, MPE is NP-complete and MAP is $\mathrm{NP}^{\mathrm{PP}}$-complete in Bayesian networks. So, how do our results compare to these results? Note that the difficulty in BNs arises from the dependencies encoded in the network, whereas for MPD and MPH, such dependencies are encoded in the query. Besides, we use the tuple-independence assumption, which makes our results different. We also note that maximal posterior probability computations have also been lifted to relational probabilistic models such as Markov Logic Networks (Richardson and Domingos 2006).

In essence, all of our data complexity results for MPD are either in polynomial time, or NP-hard. Can it be the case that MPD exhibits a *dichotomy* for universal queries (or beyond)? In other words, is it possible to achieve a complete separation between P and NP? Unfortunately, this remains open. Even though a dichotomy result has been obtained in (Gribkoff, Van den Broeck, and Suciu 2014a), this result applies only to a small fragment of universal queries. The precise classification of tractable queries is left as future work.

We have identified other interesting connections with the literature, which need to be explored further. For instance, MPH generalizes another well-studied problem in the literature, that is, the problem of finding the *shortest prime implicant* of a given propositional formula (Umans 2001). It is possible to view the most probable hypothesis as the implicant of the query; however, it does not need to be shortest in the size, but the one with the maximal probability. By adjusting the probabilities in a symmetric way, MPH can simulate the shortest prime implicant problem. Similarly, our results are closely related to propositional abductive reasoning (Eiter and Gottlob 1995). Such connections are of future interest, for obtaining more fine-grained results.

Part III

Logic and Probabilistic Knowledge Bases

Chapter 6

Ontology Languages and Query Answering

We have discussed several limitations of PDBs, already in the general introduction, and stated that the unrealistic assumptions employed in PDBs lead to even more dramatic consequences once combined with another limitation of these systems; namely, the lack of *commonsense knowledge*. Recall that the simple join query asking whether there is a composer who knows both Mozart and Beethoven,

$$\exists x\, \mathsf{Composer}(x) \wedge \mathsf{Knows}(x, \mathsf{beethoven}) \wedge \mathsf{Knows}(x, \mathsf{mozart}),$$

from the introduction is evaluated to false although the facts $\mathsf{FriendOf}(\mathsf{haydn}, \mathsf{beethoven})$, $\mathsf{TeacherOf}(\mathsf{haydn}, \mathsf{mozart})$ and $\mathsf{Composer}(\mathsf{haydn})$ are all in the knowledge base, and the given query *intuitively* follows from the given facts. Human reasoning exploits basic knowledge to deduce such implicit consequences from data; for instance, two persons who are friends know each other, or someone who is a teacher of a person knows this person and vice versa.

Clearly, we are talking about commonsense knowledge, a natural component of human reasoning, which is not present in (probabilistic) databases. A common way of encoding commonsense knowledge is in the form of ontologies. Ontologies are logical theories that formalize domain-specific knowledge, thereby allowing for automated reasoning. A popular paradigm to interpret *incomplete* data sources under commonsense knowledge in the form of ontologies is called *ontology-based data access (OBDA)* (Poggi et al. 2008).

OBDA has been very widely studied in the context of classical databases, and is also motivated by the need for open-world querying. In a nutshell, a database query is mediated by a logical interface to make implicit commonsense knowledge explicit: allowing for open-world querying, this results in more complete set of answers for the query. For example, an ontological rule can encode the commonsense knowledge in the given example and then the query will be evaluated to true under this knowledge.

In this work, we adopt the terminology from (Bienvenu, Cate, Lutz, and Wolter 2014) and speak of ontology-mediated queries (OMQs), that is, database queries (typically, unions of conjunctive queries) coupled with an ontology. The task of evaluating such queries is then called *ontology-mediated query answering (OMQA)*.

As we argued in Chapter 4, probabilistic databases are typically incomplete data sources by their nature; arguably, even more than non-probabilistic data sources. At the same time, probabilistic databases employ strong completeness assumptions, which contradict their nature. We follow a common paradigm and incorporate commonsense knowledge in probabilistic databases to alleviate these issues. Thus, this chapter is dedicated to give a brief overview on common ontology languages and introduce the paradigm of ontology-mediated query answering. We conclude this chapter by providing a summary of the known complexity results for OMQA for different ontology languages.

6.1 Ontology Languages

Ontology languages are mostly fragments of first-order logic which result from a simple trade-off. On the one hand, an ontology language has to be expressive enough to encode the domain-specific knowledge; on the other hand, the complexity of reasoning in this language should remain relatively low.

We first introduce the Datalog$^{\pm}$ family of languages, also studied under the name of *existential rules* in the literature, and then give an overview of the Description Logic (DL) family (Baader, Calvanese, et al. 2007). These formalisms, together, encompass the most-widely used knowledge representation languages in the context of ontology-based data access.

6.1.1 Datalog$^{\pm}$

The Datalog$^{\pm}$ family consists of Datalog variants, which extend plain Datalog by the possibility of existential quantification in the rule heads, and by a number of other features, and, at the same time, restrict the rule syntax in order to achieve tractability, as stated in (Calì, Gottlob, and Lukasiewicz 2012).

Vocabulary. The *domain* and the *vocabulary* of Datalog$^{\pm}$ languages are the same as for first-order logic. As a subtle difference, it is common to make a distinction between known and unknown constants in Datalog$^{\pm}$: unknown constants are used as placeholders for unknown values and are usually referred as *nulls*. Note that nulls are not considered to be part of database constants.

Datalog$^{\pm}$ Programs. We first introduce the so-called *negative constraints (NCs)*. From a database perspective, NCs can be seen as a special case of denial constraints over databases (Staworko and Chomicki 2010). Formally, a *negative constraint (NC)* is a first-order formula of the form $\forall \vec{x}\, \Phi(\vec{x}) \rightarrow \bot$, where $\Phi(\vec{x})$ is a conjunction of atoms, called the *body* of the NC, and $\bot$ is the truth constant *false*. Consider, for example, the NCs

$$\forall x\, \mathsf{Writer}(x) \wedge \mathsf{Novel}(x) \rightarrow \bot, \tag{6.1}$$

$$\forall x, y\, \mathsf{ParentOf}(x, y) \wedge \mathsf{ParentOf}(y, x) \rightarrow \bot. \tag{6.2}$$

The former states that *writers* and *novels* are disjoint entities, whereas the latter asserts that the ParentOf relation is antisymmetric.

To formulate more general ontological knowledge, tuple-generating dependencies are introduced. Intuitively, such dependencies describe constraints on databases in the form of generalized Datalog rules with existentially quantified conjunctions of atoms in rule heads. Formally, a *tuple-generating dependency (TGD)* is a first-order formula of the form $\forall \vec{x}\, \Phi(\vec{x}) \rightarrow \exists \vec{y}\, \Psi(\vec{x}, \vec{y})$, where $\Phi(\vec{x})$ is a conjunction of atoms, called the *body* of the TGD, and $\Psi(\vec{x}, \vec{y})$ is a conjunction of atoms, called the *head* of the TGD. Consider, for example, the TGDs

$$\forall x, y\, \mathsf{AuthorOf}(x, y) \wedge \mathsf{Novel}(y) \rightarrow \mathsf{Writer}(x), \tag{6.3}$$

$$\forall y\, \mathsf{Novel}(y) \rightarrow \exists x\, \mathsf{AuthorOf}(x, y) \wedge \mathsf{Writer}(x). \tag{6.4}$$

The first one states that anyone who authors a novel is a writer. The second one asserts that all novels are authored by a writer. Note that TGDs can express the well-known inclusion dependencies and join dependencies from database theory. A *Datalog$^{\pm}$ program* (or *ontology*) Σ is a finite set of negative constraints and tuple generating dependencies. A Datalog$^{\pm}$ program is *positive* if it consists of only TGDs, i.e., does not contain any NCs.

Semantics. In essence, Datalog$^{\pm}$ languages are only syntactic fragments of first-order logic, which also employ the *standard name assumption*, as in databases. Thus, a first-order interpretation $\mathcal{I}$ is a model of an ontology Σ in the classical sense, i.e., if $\mathcal{I} \models \alpha$ for all $\alpha \in \Sigma$. Given a database $\mathcal{D}$ defined over known constants and an ontology Σ, we write $mods(\Sigma, \mathcal{D})$ to represent the set of models of Σ that extend $\mathcal{D}$, which is formally defined as $\{\mathcal{I} \mid \mathcal{I} \models \mathcal{D}, \mathcal{I} \models \Sigma\}$. As a consequence, a database $\mathcal{D}$ is *consistent* w.r.t. a Σ if $mods(\Sigma, \mathcal{D})$ is non-empty.

Overview of the Datalog$^{\pm}$ Family. As we will discuss later, the entailment problem in Datalog$^{\pm}$ ontologies is undecidable (Beeri and Vardi 1981), which motivated syntactic restrictions on Datalog$^{\pm}$ ontologies (Baget, Mugnier, Rudolph, and Thomazo 2011; Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012; Fagin, Kolaitis, Miller, and Popa 2005; Krötzsch and Rudolph 2011).

There are a plethora of classes of TGDs; here, we only recall some basic classes. The most important (syntactic) restrictions on TGDs studied in the literature are guardedness (Calì, Gottlob, and Kifer 2013), stickiness (Calì, Gottlob, and Pieris 2012) and acyclicity, along with their "weak" counterparts, weak guardedness (Calì, Gottlob, and Kifer 2013), weak stickiness (Calì, Gottlob, and Pieris 2012), and weak acyclicity (Fagin, Kolaitis, et al. 2005), respectively.

A TGD is *guarded*, if there exists a body atom that contains (or "guards") all body variables. 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 *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 (Calì, Gottlob, and Kifer 2013) for full details). The associated class of TGDs is denoted WG. It is easy to verify that $\mathsf{L} \subset \mathsf{G} \subset \mathsf{WG}$.

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 TGD 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 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. A set of TGDs 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 $\mathsf{WA} \subset \mathsf{WS}$ (Calì, Gottlob, and Pieris 2012).

Another key fragment of TGDs which deserves our attention are *full* TGDs, i.e., TGDs without existentially quantified variables. The corresponding class is denoted by F. Restricting full TGDs to satisfy linearity, guardedness, stickiness, or acyclicity yields the classes LF, GF, SF, and AF, respectively. It is known that $\mathsf{F} \subset \mathsf{WA}$ (Fagin, Kolaitis, et al. 2005) and $\mathsf{F} \subset \mathsf{WG}$ (Calì, Gottlob, and Kifer 2013).

Other classes of TGDs, which are not central to our analysis include *tame* (T); *frontier-guarded* (FG), *weakly frontier-guarded* (WFG); *sticky-join* (SJ); *weakly sticky-join* (WSJ); *jointly acyclic* (JA); *frontier-one* (F1); *jointly guarded* (JG); *glut guarded* (GG); *glut frontier guarded* (GFG). The inclusion relationships between these classes relative to their expressiveness are given as follows.

$$\begin{aligned} \mathsf{F1} \subset \mathsf{FG}, \quad \mathsf{WG} \subset \mathsf{WFG} \subset \mathsf{GFG}, \quad \mathsf{WG} \subset \mathsf{JG} \subset \mathsf{GG}, \\ \mathsf{WA} \subset \mathsf{JA} \subset \mathsf{GFG}, \quad \mathsf{JA} \subset \mathsf{GG}, \quad \mathsf{WS} \subset \mathsf{WSJ}, \\ \mathsf{S} \subset \mathsf{SJ}, \quad \mathsf{L} \subset \mathsf{SJ} \subset \mathsf{WSJ}, \quad \mathsf{S} \subset \mathsf{T}, \mathsf{G} \subset \mathsf{T}. \end{aligned}$$

For details on these classes, we refer to the relevant literature (Baget et al. 2011; Calì, Gottlob, and Pieris 2012; Gottlob, Manna, and Pieris 2013, 2014; Gottlob, Rudolph, and Simkus 2014; Krötzsch and Rudolph 2011).

Notation. We usually omit the universal quantifiers in TGDs and NCs, and for clarity we consider single-atom-head TGDs; however, our results can be easily extended to TGDs with conjunctions of atoms in the head (except under the bounded-arity assumption). Following the common convention, we will assume that NCs are part of all Datalog$^{\pm}$ languages. When we focus only on positive programs, we make this explicit by annotating the respective language with "+"; for example, G^{+} denotes the class of guarded programs which do not contain any negative constraints.

Systems and Applications. RDFox (Nenov, Piro, Motik, Horrocks, Wu, and Banerjee 2015) is a highly scalable Datalog engine. Note that many of the Datalog$^{\pm}$ languages can be encoded into a (possibly large) Datalog program. Linear, guarded, and sticky-join languages of Datalog$^{\pm}$ are also supported by the Nyaya knowledge management system (De Virgilio, Orsi, Tanca, and Torlone 2011). Traditionally, Datalog has been used as a query language for databases. The most prominent application of Datalog$^{\pm}$ languages are in the context of ontology-mediated query answering which we will discuss later.

6.1.2 Description Logics

Description Logics (DLs) is a large family of logical formalisms that aim to combine expressivity and efficient reasoning (Baader, Calvanese, et al. 2007). There are many DLs proposed in the literature, ranging from the inexpressive $\mathcal{EL}$ (Baader, Brandt, and Lutz 2005) and *DL-Lite* (Artale, Calvanese, Kontchakov, and Zakharyaschev 2009), over the prototypical expressive $\mathcal{ALC}$ (Schmidt-Schauß and Smolka 1991), to the very expressive $\mathcal{SROIQ}$ (Horrocks, Kutz, and Sattler 2006). We provide an overview of these languages.

Table 6.1: Syntax and semantics of DL concept and role constructors, TBox axioms, and ABox assertions. The (dual) constructors bottom ($\bot$), disjunction ($\sqcup$), value restriction ($\forall$), and at-least restrictions ($\geq$) are omitted from the table. As usual, we use A, B to denote concept names and C, D to denote (complex) concepts.

Name	Syntax	Semantics	
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	
individual name	a	$a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$	Names
top	$\top$	$\Delta^{\mathcal{I}}$	
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	
existential restr.	$\exists r.C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists\, e \in C^{\mathcal{I}} :\ (d,e) \in r^{\mathcal{I}}\}$	
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$	
at-most restr.	$\leq n\, r.C$	$\{d \in \Delta^{\mathcal{I}} \mid \sharp\{e \in C^{\mathcal{I}} \mid (d,e) \in r^{\mathcal{I}}\} \leq n\}$	Concepts
inverse role	r^-	$\{(e,d) \mid (d,e) \in r^{\mathcal{I}}\}$	Roles
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	
role inclusion	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$	
transitivity axiom	$trans(r)$	$r^{\mathcal{I}} = (r^{\mathcal{I}})^+$	TBox axioms
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	
role assertion	$r(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$	ABox assertions

Vocabulary. In DLs, a domain of interest is described via a vocabulary consisting of three countably infinite, pairwise disjoint sets of symbols: the set $\mathsf{N_C}$ of *concept names*, to capture classes of objects, and the set $\mathsf{N_R}$ of *role names*, to capture binary relations between objects, the set $\mathsf{N_I}$ of *individual names* to refer to specific individual objects. A DL vocabulary can therefore be seen as a restricted *first-order logic* vocabulary containing only *unary predicates* (concept names), *binary predicates* (role names), and *constants* (individual names).

Concept Language. The basic building blocks of the DL syntax are concepts that represent sets of objects (e.g., Human), roles relating objects to objects via binary relations (e.g., hasParent), and individuals representing concrete objects (e.g. alice). DL *concepts* are built inductively from concept and role names, using the concept constructors given in Table 6.1. The expressiveness of the resulting DL depends on the concept constructors that are allowed in the language and on other syntactic restrictions being posed.

Knowledge Base. These concepts are then used to form the axioms and assertions of the respective language, which are in the form of *general concept inclusions (GCIs)*, *role*

inclusions, *transitivity axioms*, *concept assertions*, or *role assertions* as given in Table 6.1. A *TBox axiom* is either a GCI, or a *role inclusion*, or a *transitivity axiom*. An *ABox axiom, or assertion* is either a *concept assertion*, or a *role assertion*. For example, the NC (6.1) and NC (6.2) can be encoded in DLs in terms of the GCIs

$$\mathsf{Writer} \sqsubseteq \neg\mathsf{Novel} \qquad \textit{and} \qquad \mathsf{ParentOf} \sqsubseteq \neg\mathsf{ParentOf}^{-},$$

respectively. Similarly, the TGD (6.3) and TGD (6.4) can also be represented as GCIs:

$$\exists\,\mathsf{AuthorOf} \sqcap \mathsf{Novel} \sqsubseteq \mathsf{Writer} \qquad \textit{and} \qquad \mathsf{Novel} \sqsubseteq \exists\,\mathsf{AuthorOf}^{-}.\mathsf{Writer}.$$

The following self-explaining facts

$$\mathsf{Writer}(\mathsf{balzac}) \qquad \textit{and} \qquad \mathsf{AuthorOf}(\mathsf{hamsun}, \mathsf{hunger}).$$

are examples of concept and role assertions, respectively. Note that traditionally roles are written in lowercase letters in DLs; here, we do not follow this convention for the sake of consistency with Datalog$^{\pm}$ notation.

Finally, a *TBox*, or an *ontology*, is a finite set of axioms, an *ABox* is a finite set of assertions. Then, a knowledge base is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox, and $\mathcal{A}$ is an ABox.

Semantics. The *semantics* of DLs is based on first-order *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where every concept name A is mapped to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and every role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. This interpretation function is extended to arbitrary concepts and axioms as given in the right-most column of Table 6.1. Dual concept constructors such as *bottom* ($\bot$), *disjunction* ($\sqcup$), *universal restriction* ($\forall$), and *at-least restriction* ($\geq$) can be simulated by the given ones and thus are omitted from the table. Clearly, the semantics can equivalently be given by a translation to first-order logic (with equality) as in (Borgida 1996). We prefer to define the semantics directly as the translation to first-order logic becomes somehow more intricate due to the number restrictions.

We define an interpretation $\mathcal{I}$ to be *a model of a TBox* $\mathcal{T}$, denoted $\mathcal{I} \models \mathcal{T}$, if it satisfies all the axioms in $\mathcal{T}$. Similarly, $\mathcal{I}$ *is a model of an ABox* $\mathcal{A}$, denoted $\mathcal{I} \models \mathcal{A}$, if it satisfies all the assertions in $\mathcal{A}$. Finally, $\mathcal{I}$ *is a model of a KB* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, denoted $\mathcal{I} \models \mathcal{K}$, if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$.

Standard Reasoning Problems. Classical reasoning tasks in DLs are ontology consistency, i.e., checking whether the ontology is non-contradictory; concept satisfiability, i.e., checking whether a given concept is non-contradictory in the ontology; subsumption, i.e., checking whether a specific concept is included in a more general one; instance checking, i.e., checking whether a new assertion follows from the ontology. We refer the reader to the relevant literature for the details (Baader, Calvanese, et al. 2007). Our concern is mostly on ontology-mediated query answering, which will be introduced in the next section. Besides, it is well-known that most of these classical reasoning tasks can be reduced to ontology-mediated query answering.

Overview of the DL Family. We are concerned with a variety of different DLs that are all well-known from the literature. For instance, the basic DL $\mathcal{ALC}$ allows for GCIs as TBox axioms, and the concept language of $\mathcal{ALC}$ allows only for the constructors *top*, *bottom*, *negation*, *conjunction*, *disjunction*, *existential restrictions*, and *value restrictions*. The DL $\mathcal{S}$ extends $\mathcal{ALC}$ with *transitivity axioms*. Traditionally, DLs are named relative to the concept constructors and the types of axioms allowed in the language. For instance, the letters $\mathcal{H}$, $\mathcal{I}$, $\mathcal{O}$, and $\mathcal{Q}$ denote *role inclusions*, *inverse roles*, *nominals*, and *(qualified) number restrictions*, respectively. Depending on what is allowed in the language, we obtain a variety of DLs, such as the very expressive DL $\mathcal{SHOIQ}$, which allows all the constructors given in Table 6.1 (while restricting the interaction of number restrictions and transitive roles). For further details on expressive DLs, we refer to (Tobies 2001).

A special emphasis is put on the so-called *light-weight DLs* in the literature. In particular, the inexpressive DLs based on $\mathcal{EL}$ (Baader, Brandt, and Lutz 2005) and *DL-Lite* (Artale, Calvanese, et al. 2009) are very widely studied due to their nice computational properties. $\mathcal{EL}$ is a sub-logic of $\mathcal{ALC}$ which only allows the constructors *top*, *conjunction*, and *existential restrictions*. The *DL-Lite* family also does not allow for value restrictions, while inverse roles are allowed and existential restrictions can only be of the form $\exists r.\top$ and are thus abbreviated as $\exists r$. Moreover, the core dialect of *DL-Lite* allows only for concept inclusions of the form $B_1 \sqsubseteq B_2$ and $B_1 \sqsubseteq \neg B_2$, where B_1 and B_2 are either concept names or of the form $\exists r$. One famous dialect of this family is *DL-Lite*$_\mathcal{R}$, which additionally allows for a restricted type of role inclusions. For further details and other dialects of the *DL-Lite* family, we refer the reader to (Artale, Calvanese, et al. 2009).

Systems and Applications. Applications of DLs range from standardization efforts like the Web Ontology Language OWL 2 (Group 2012; Horrocks, Patel-Schneider, and Harmelen 2003) to the formalization of many kinds of domain knowledge. DLs have been successfully employed for creating large knowledge bases, representing real application domains. For instance, they are the logical formalism underlying prominent bio-medical ontologies such as the large biomedical ontology SNOMED CT[1], or the knowledge base GALEN[2], or the GENE ONTOLOGY (Ashburner et al. 2000).

Efficient algorithms are implemented for expressive DLs, which resulted in a number of reasoners such as HermiT (Glimm, Horrocks, Motik, Stoilos, and Wang 2014), Konclude (Steigmiller, Liebig, and Glimm 2014), FaCT++ (Tsarkov and Horrocks 2006), and Pellet (Sirin, Parsia, Grau, Kalyanpur, and Katz 2007). These reasoners support most of the standard reasoning tasks, while Pellet also partially supports OMQA. The more recent reasoner PAGOdA (Zhou, Nenov, Grau, and Horrocks 2014) is tailored for OMQA. Special purpose systems for reasoning with lightweight DLs have also been developed. ELK (Kazakov, Krötzsch, and Simančík 2012) is a reasoner specifically tailored for the DL $\mathcal{EL}$ and its known extensions, which also focuses on classical reasoning tasks. On the other hand, the *DL-Lite* reasoner Mastro fully supports OMQA (Calvanese, De Giacomo, Lembo, Lenzerini, Poggi, Rodriguez-Muro, Rosati, Ruzzi, and Savo 2011).

[1] `http://www.ihtsdo.org/snomed-ct/`

[2] `https://bioportal.bioontology.org/ontologies/GALEN`

Table 6.2: A database that consists of the unary relations Writer, Novel and the binary relation AuthorOf.

Writer	AuthorOf		Novel
balzac	hamsun	hunger	goriot
dostoyevski	dostoyevski	gambler	hunger
kafka	kafka	trial	trial

6.2 Ontology-Mediated Query Answering

Ontology-mediated query answering is a popular paradigm for querying incomplete data sources in a more adequate manner (Bienvenu, Cate, et al. 2014). Formally, an *ontology-mediated query (OMQ)* is a pair $(Q, \mathcal{T})$, where Q is a Boolean query and $\mathcal{T}$ is an ontology.

Given a database $\mathcal{D}$ and an OMQ $(Q, \mathcal{T})$, we say that $\mathcal{D}$ entails the OMQ $(Q, \mathcal{T})$, denoted $\mathcal{D} \models (Q, \mathcal{T})$, if for all models $\mathcal{I} \models (\mathcal{T}, \mathcal{D})$ it holds that $\mathcal{I} \models Q$. Then, *ontology-mediated query answering (OMQA)* is the task of deciding whether $\mathcal{D} \models (Q, \mathcal{T})$ for a given database $\mathcal{D}$ and an OMQ $(Q, \mathcal{T})$. Note that we use the term query answering in a rather loose sense to refer to the Boolean query evaluation problem.

In this work, we mostly restrict our attention to OMQs which contain unions of conjunctive queries. Nevertheless, we sometimes also refer to *instance queries*, which is mostly of interest in the context of DLs. Formally, an instance query is a unary atom $C(x)$ and *instance query answering* is to find all ABox individuals a_i,$1 \leq i \leq n$, such that $C(a_i)$ is entailed by the knowledge base. The corresponding decision problem is typically defined on atomic queries, that is, unary, ground atoms. Given an atomic query, *instance checking*, is to check whether the atom is entailed by the knowledge base.

Obviously, these definitions apply to DL ontologies as well as to Datalog$^{\pm}$ ontologies. Note, however, that classical DLs consists of unary and binary predicates only. Therefore, it is common to assume that the database is preprocessed and transformed into an ABox, which intuitively represents an abstraction of the database over the vocabulary of the ontology. We will therefore replace the database with an ABox in DLs.

We now briefly illustrate these notions on the simple database that consists of writers (Writer) and novels (Novel) and the relation AuthorOf, as shown in Table 6.2.

Example 6.1 Let us consider the database $\mathcal{D}_a$ given in Table 6.2. Observe that the simple queries

$$Q_1 \coloneqq \mathsf{Writer}(\mathsf{hamsun}) \quad \textit{and} \quad Q_2 \coloneqq \exists x\, \mathsf{Writer}(x) \wedge \mathsf{AuthorOf}(x, \mathsf{goriot}),$$

are not satisfied by the database $\mathcal{D}_a$ although they should evaluate to true from an intuitive perspective. On the other side, under the Datalog$^{\pm}$ program Σ_a that consists of the TGDs

$$\forall x, y\, \mathsf{AuthorOf}(x, y) \wedge \mathsf{Novel}(y) \rightarrow \mathsf{Writer}(x), \tag{6.3}$$

$$\forall y\, \mathsf{Novel}(y) \rightarrow \exists x\, \mathsf{AuthorOf}(x, y) \wedge \mathsf{Writer}(x), \tag{6.4}$$

both of these queries are satisfied: $\mathcal{D}_a \models (Q_1, \Sigma_a)$ holds due to the first rule and $\mathcal{D}_a \models (Q_2, \Sigma_a)$ holds due to the second rule. The incomplete database is queried through

Table 6.3: Complexity of ontology-mediated query answering in Datalog$^{\pm}$.

Languages	data	fixed-program	bounded-arity	combined
L, LF, AF	in AC^0	NP	NP	PSPACE
S, SF	in AC^0	NP	NP	EXP
A	in AC^0	NP	NEXP	NEXP
F, GF	P	NP	NP	EXP
G	P	NP	EXP	2EXP
WS, WA	P	NP	2EXP	2EXP
WG	EXP	EXP	EXP	2EXP
F1, FG	P	NP	2EXP	2EXP
WFG	EXP	EXP	2EXP	2EXP
JA	P	NP	2EXP	2EXP
JG	EXP	EXP	EXP–2EXP	2EXP
JFG	EXP	EXP	2EXP	2EXP
GG	EXP	EXP	3EXP	3EXP
GFG	EXP	EXP	3EXP	3EXP
SJ	in AC^0	NP	NP–EXP	EXP
WSJ	P	NP	2EXP	2EXP
T	P	NP	EXP	2EXP

the logical rules that encode commonsense knowledge, which in turn results in more complete answers. ◊

A key paradigm in ontology-mediated query answering is the *first-order rewritability* of queries, which we introduce next. Intuitively, FO-rewritability ensures that we can rewrite an OMQ into a (possibly large) UCQ and this transformation is *homomorphism preserving over all finite structures* (Rossman 2008).

Definition 6.2 (FO-rewritability) Let $\mathcal{T}$ be an ontology and Q a Boolean query. Then, the OMQ $(Q, \mathcal{T})$ is *FO-rewritable* if there exists a Boolean UCQ $Q_{\mathcal{T}}$ such that, for all databases $\mathcal{D}$ that are consistent w.r.t. $\mathcal{T}$, we have $\mathcal{D} \models (Q, \mathcal{T})$ if and only if $\mathcal{D} \models Q_{\mathcal{T}}$. In this case, $Q_{\mathcal{T}}$ is called an *FO-rewriting* of $(Q, \mathcal{T})$. A language $\mathcal{L}$ is *FO-rewritable* if it admits an FO-rewriting for any UCQ and theory in $\mathcal{L}$. ◊

FO-rewritability is important since it implies a data-independent reduction from OMQA to query evaluation in relational databases. In practical terms, this means that the query can be rewritten into an SQL query to be evaluated in relational database management systems. In theoretical terms, this puts OMQA in AC^0 in data complexity for all FO-rewritable languages. In what follows, we summarize the known complexity results for OMQA in Datalog$^{\pm}$ and then in Description Logics.

6.2.1 Ontology-Mediated Query Answering in Datalog$^{\pm}$

As before, we study the data, bounded-arity and combined complexity: *Data complexity* is calculated based on the database; i.e., the program and the query are assumed to be fixed (Vardi 1982), and the *combined complexity* is calculated by considering all

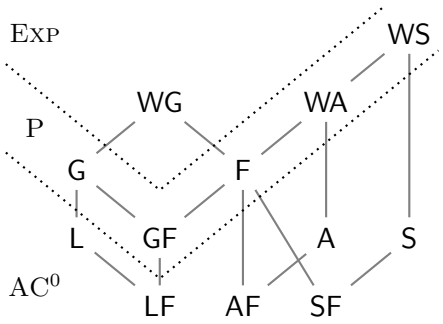


Figure 6.1: Inclusion relationships and data complexity of ontology-mediated query answering for Datalog$^{\pm}$ languages

components. In the *bounded-arity complexity*, we additionally assume that the arity of the predicates is bound by some integer value. We additionally consider *fixed-program complexity* in Datalog$^{\pm}$, which is calculated in the size of the database and the query, while the program remains fixed.

The complexity of ontology-mediated query answering in Datalog$^{\pm}$ is summarized in Table 6.3, which is organized in five main blocks. Many of our results are of a generic nature; thus, we prefer to focus on a representative set of languages (i.e., the first block of languages) for ease of presentation. Note, however, that our results can be generalized to other classes in a rather straight-forward manner.

Datalog$^{\pm}$ consists of Datalog variants; thus, the complexity of evaluating Datalog queries is central to the results given in Table 6.3. It is known that evaluating plain Datalog queries is in P in data complexity and EXP-complete in combined complexity (Immerman 1986; Vardi 1982).

The complexity results given in the first block are from (Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012; Calì, Gottlob, and Pieris 2012; Fagin, Kolaitis, et al. 2005; Lukasiewicz, Martinez, Pieris, and Simari 2015). The inclusion relationships between these classes, organized relative to their data complexity is depicted in Figure 6.1; many of these classes such as L, A and S are FO-rewritable and thus OMQA is in AC^0 for these classes. Notice that the class GF is already not FO-rewritable. GF is the least expressive language in this block among the languages with P-complete data complexity. The only class that is intractable in data complexity is WG, for which OMQA is already EXP-complete.

OMQA is already NP-hard for all classes in fixed-program complexity, which immediately follows from NP-hardness of conjunctive query evaluation in databases (Chandra and Merlin 1977). Bounded-arity complexity results of Datalog$^{\pm}$ languages are closely linked to those in DLs (which allow only unary and binary predicates). The complexity of OMQA is either the same or higher in bounded-arity complexity compared to fixed-program complexity. From a theoretical perspective, the class A is of interest, because it has a nondeterministic bounded-arity complexity, which leads to a different behavior, in general. In the last column, the combined complexity results are given, which are higher than the bounded-arity complexity in all cases, except for the classes A, WS and WA.

Table 6.4: Complexity of ontology-mediated (instance) query answering in different DLs. All results without $\leq$ and $\geq$ denote completeness results. We note that no elementary upper bound is known for ontology-mediated query answering in $\mathcal{SHOIQ}$.

Description Logic	Instance Queries		Unions of Conjunctive Queries	
	data	combined	data	combined
DL-Lite, *DL-Lite*$_\mathcal{R}$	$\leq$ AC^0	NLogSpace	$\leq$ AC^0	NP
$\mathcal{EL}$, $\mathcal{ELH}$	P	P	P	NP
$\mathcal{ELI}$, Horn-$\mathcal{SHOIQ}$	P	Exp	P	Exp
$\mathcal{ALC}$, $\mathcal{ALCHQ}$	coNP	Exp	coNP	Exp
$\mathcal{ALCI}$, $\mathcal{SH}$, $\mathcal{SHIQ}$	coNP	Exp	coNP	2Exp
$\mathcal{SHOIQ}$	coNP	coNExp	$\geq$ coNP	$\geq$ coN2Exp

We conclude by noting the sources for the complexity results in the other blocks. Results given in the second block of Table 6.3 are mostly from (Baget et al. 2011; Gottlob, Rudolph, and Simkus 2014); the third block from (Gottlob, Rudolph, and Simkus 2014; Krötzsch and Rudolph 2011); the fourth block from (Calì, Gottlob, and Pieris 2012; Gottlob, Manna, and Pieris 2014) and tame class is investigated in (Gottlob, Manna, and Pieris 2013, 2014)

6.2.2 Ontology-Mediated Query Answering in Description Logics

The relative expressivity of DL languages depends on various factors, as explained before. This results in a rich map for the computational complexity for OMQA; see especially (Bienvenu and Ortiz 2015) for a tutorial on OMQA based on DLs. Table 6.4 summarizes the results for ontology-mediated *instance* query answering and ontology-mediated query answering. We will refer these problems simply as query answering and instance query answering, respectively.

To start with, we note that instance query answering can be reduced to standard reasoning tasks such as consistency for languages that include conjunction and negation as concept constructor (Baader, Calvanese, et al. 2007). As a consequence, most of the results related to instance query answering follow from the complexity of standard reasoning tasks, which we use in the sequel.

The light-weight DLs *DL-Lite* and *DL-Lite*$_\mathcal{R}$ are FO-rewritable, that is, query answering for these languages is in AC^0 in data complexity. As for the combined complexity, standard reasoning in these languages is shown to be NLogSpace-complete, which coincides with the complexity of instance query answering. In contrast, query answering is NP-complete in combined complexity. For detailed insights, see (Artale, Calvanese, et al. 2009; Calvanese, De Giacomo, Lembo, Lenzerini, and Riccardo Rosati 2007) and the references therein.

The data tractability results for $\mathcal{EL}$, $\mathcal{ELH}$ and $\mathcal{ELI}$ follow from (Krisnadhi and Lutz 2007; Krötzsch and Rudolph 2007; Ricardo Rosati 2007). In combined complexity, the DL $\mathcal{ELI}$ and the DLs $\mathcal{EL}$ and $\mathcal{ELH}$ behave rather differently for query answering. Instance query answering for $\mathcal{EL}$ and $\mathcal{ELH}$ is in polynomial time, while this problem is

EXP-complete for $\mathcal{ELI}$ (Baader, Brandt, and Lutz 2005). Furthermore, query answering for $\mathcal{EL}$ and $\mathcal{ELH}$ is NP-complete, while it is EXP-complete for $\mathcal{ELI}$. A closely related logic is Horn-$\mathcal{SHIQ}$, for which query answering is shown to be P-complete in data and EXP-complete in combined complexity (Eiter, Gottlob, Ortiz, and Šimkus 2008); this result is then extended to Horn-$\mathcal{SHOIQ}$ (Ortiz, Rudolph, and Šimkus 2011).

All the remaining logics from Table 6.4 are beyond the so-called Horn fragment; among these, the least expressive one is the DL $\mathcal{ALC}$, for which (instance) query answering is CONP-complete in data complexity as shown in (Calvanese, De Giacomo, Lembo, Lenzerini, and Riccardo Rosati 2013) and the upper bound holds for logics up to $\mathcal{SHIQ}$ (Glimm, Horrocks, Lutz, and Sattler 2008). Standard reasoning tasks in $\mathcal{ALC}$ are already EXP-complete in combined complexity (Schild 1991). This explains the lower bounds for query answering in $\mathcal{ALC}$ and upper bounds hold even for $\mathcal{SHQ}$ in combined complexity (Lutz 2008). For $\mathcal{ALCI}$, which extends $\mathcal{ALC}$ with inverse roles, query answering is already 2EXP-hard as shown in (Lutz 2008) and this is a matching lower bound since query answering in $\mathcal{SHIQ}$ is in 2EXP (Glimm, Horrocks, Lutz, et al. 2008)

Standard reasoning in $\mathcal{SHOIQ}$ is CONP complete in the data and CONEXP-complete in combined complexity, which transfer to instance query answering. Unfortunately only a CON2EXP-hardness is known for query answering in $\mathcal{SHOIQ}$ in combined complexity (Glimm, Kazakov, and Lutz 2011).

Chapter 7

Probabilistic Data Access with Datalog$^{\pm}$

Our proposal is to incorporate commonsense knowledge in probabilistic databases to achieve better means of querying of incomplete sources. This is inherently connected to giving up the completeness assumptions of standard probabilistic databases, i.e., the CWA and the probabilistic independence of facts. For example, encoding the knowledge that two persons who are friends know each other in the form of ontological knowledge essentially induces dependencies on probabilistic facts and also allows us to deduce facts that are not in the database. In essence, we are talking about *probabilistic knowledge bases*, which in addition to a probabilistic database contain a logical theory.

At this point, it is important to note that the independence assumption of probabilistic databases is important for efficiency reasons. In other words, one can allow dependencies directly on the data, but this could then easily result in the loss of efficiency for a large class of queries. Our goal is, on the one hand, to relax these assumptions by introducing dependencies on a first-order level, and on the other hand, possibly leaving a large amount of data still probabilistically independent. Whenever possible, we will also quest for lifting the data complexity dichotomy results of probabilistic databases to the case of ontology-mediated queries.

From a broader perspective, extending tuple-independent probabilistic data models with logical rules is an old idea, aiming to induce correlations on a logical level, and thus to relax the tuple independence, resulting in very powerful formalisms (Poole 1997). As already pointed out in the general introduction, recent approaches are based on MLNs (Gribkoff and Suciu 2016a) and are thus based on the closed-domain assumption. Our results are based on well-known ontology languages, which allow fully fledged first-order knowledge with a proper handling of anonymous individuals.

In this chapter, we extend the problems introduced in Part II to the case of ontology-mediated queries based on Datalog$^{\pm}$ languages (Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012; Calì, Gottlob, and Pieris 2012; Fagin, Kolaitis, et al. 2005; Lukasiewicz, Martinez, et al. 2015). Therefore, this chapter is organized in three main sections in correspondence with chapters 3, 4 and 5, respectively.

First, we extend our results from Chapter 3 and study OMQ evaluation for PDBs. We discuss how PDBs can benefit from an explicit encoding of ontological knowledge. Afterwards, we study OMQ evaluation for OpenPDBs; thus extend our results from Chapter 4. In the presence of ontological rules, both upper and lower probabilities of queries become even more informative in OpenPDBs, enabling us to distinguish queries that were indistinguishable before. Finally, we revisit the most probable database and most probable hypothesis problems from Chapter 5 in the context of OMQs. We conclude with an overview of the related work and the results.

7.1 Ontology-Mediated Queries for Probabilistic Databases

We study the problem of evaluating ontology-mediated queries for PDBs. In a nutshell, we allow Datalog$^{\pm}$ programs on top of tuple-independent PDBs and are interested in the probability of a given OMQ. The semantics extends the classical OMQA with the possible worlds semantics.

Definition 7.1 (semantics) The *probability of an OMQ* (Q, Σ) *relative to a probability distribution* P is

$$\mathrm{P}(Q, \Sigma) = \sum_{\mathcal{D} \models (Q,\Sigma)} \mathrm{P}(\mathcal{D}),$$

where $\mathcal{D}$ ranges over all databases over σ. $\Diamond$

The major difference compared to PDBs is that this semantics defers the decision of whether a world satisfies a query to an entailment test, which also includes a logical theory.

Note that the Datalog$^{\pm}$ program can be inconsistent with some of the worlds $\mathcal{D}$ induced by the PDB. The fact that a program contains an inconsistency makes standard reasoning very problematic, as *anything* can be entailed from an inconsistent theory (*"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.

The common way of tackling this problem for probabilistic knowledge bases is to restrict the probabilistic query evaluation 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 formally, the query probabilities can be normalized by defining

$$\mathrm{P}_n(Q, \Sigma) := (\mathrm{P}(Q, \Sigma) - \gamma)/(1 - \gamma),$$

where γ is the probability of the inconsistent worlds given as

$$\gamma := \sum_{mods(\Sigma,\mathcal{D})=\emptyset} \mathrm{P}(\mathcal{D}).$$

The normalization factor γ can therefore be computed once and then reused as a post-processing step. Therefore, for simplicity, throughout this section, we assume that all the worlds $\mathcal{D}$ induced by the PDB are consistent with the program, in which case we also say that the PDB (or the probability distribution induced) is consistent with the program.

Let us now illustrate the the effect of ontological rules in PDBs. Recall Example 6.1, where we illustrated the effect of ontological rules in querying databases. Queries which intuitively follow from the knowledge encoded in the database were not satisfied by the given database (from Table 6.2). Still, it was possible to alleviate this problem using ontological rules. We now adopt this example to PDBs and observe a similar effect.

Table 7.1: A probabilistic database $\mathcal{P}_a$ which consists of the unary relations Writer, Novel and the binary relation AuthorOf.

Writer	P
balzac	0.8
dostoyevski	0.6
kafka	0.9

AuthorOf		P
hamsun	hunger	0.9
dostoyevski	gambler	0.6
kafka	trial	0.8

Novel	P
goriot	0.7
hunger	0.4
trial	0.5

Example 7.2 Let us consider the PDB $\mathcal{P}_a$ given in Table 7.1. The queries

$$Q_1 \coloneqq \mathsf{Writer}(\mathsf{hamsun}) \quad \textit{and} \quad Q_2 \coloneqq \exists x\, \mathsf{Writer}(x) \wedge \mathsf{AuthorOf}(x, \mathsf{goriot}),$$

from Example 6.1 evaluate to the probability 0 on $\mathcal{P}_a$. On the other side, under the Datalog$^{\pm}$ program Σ_a that consists of the rules

$$\forall x, y\, \mathsf{AuthorOf}(x, y) \wedge \mathsf{Novel}(y) \rightarrow \mathsf{Writer}(x),$$
$$\forall y\, \mathsf{Novel}(y) \rightarrow \exists x\, \mathsf{AuthorOf}(x, y) \wedge \mathsf{Writer}(x),$$

there are worlds where both of these queries are satisfied. One such world is $\mathcal{D}_a$ given in Table 6.2, i.e., recall that $\mathcal{D}_a \models (Q_1, \Sigma_a)$ and $\mathcal{D}_a \models (Q_2, \Sigma_a)$. More precisely, we obtain that $\mathrm{P}(Q_1) = 0.63$ since any world which contains both $\mathsf{AuthorOf}(\mathsf{hamsun}, \mathsf{hunger})$ and $\mathsf{Novel}(\mathsf{hunger})$, entails $\mathsf{Writer}(\mathsf{hamsun})$ and no other world does. Similarly, $\mathrm{P}(Q_2) = 0.7$ since Q_2 is entailed from all and only those worlds where $\mathsf{Novel}(\mathsf{goriot})$ holds. ◊

Probabilistic query evaluation, as a decision problem, is defined as in Chapter 3 with the only difference that it is now parametrized with ontology-mediated queries.

Definition 7.3 (probabilistic query evaluation) Given a PDB $\mathcal{P}$, an OMQ (Q, Σ) and a value $p \in [0, 1)$, *probabilistic query evaluation*, denoted PQE, is to decide whether $\mathrm{P}_{\mathcal{P}}(Q, \Sigma) > p$ holds. PQE is parametrized with the language of the ontology and the query; we write $\mathsf{PQE}(\mathsf{Q}, \mathcal{L})$ to define PQE on the class Q of queries and on the class of ontologies restricted to the language $\mathcal{L}$. ◊

We will deliberately use the terms probabilistic query evaluation and probabilistic OMQ evaluation interchangeably if there is no danger of ambiguity. We now provide a host of complexity results for probabilistic OMQ evaluation relative to different languages.

7.1.1 Complexity Results

We start our complexity analysis with a rather simple result that is of a generic nature. Intuitively, given the complexity of OMQA in a Datalog$^{\pm}$ language, we obtain an immediate upper and lower bound for the complexity of probabilistic OMQ evaluation in that language.

Theorem 7.4 *Let $\mathfrak{C}$ denote the data (respectively, fixed-program, bounded-arity, combined) complexity of ontology-mediated query answering for a Datalog$^{\pm}$ language $\mathcal{L}$. Then, $\mathsf{PQE}(\mathsf{UCQ}, \mathcal{L})$ is $\mathfrak{C}$-hard and in $\mathrm{PP}^{\mathfrak{C}}$ for PDBs in data (respectively, fixed-program, bounded-arity, combined) complexity.*

Proof. Let (Q, Σ) be an OMQ, $\mathcal{P}$ be a PDB, and $p \in [0, 1)$ be a threshold value. Consider a nondeterministic Turing machine with a $\mathfrak{C}$ oracle. Similar to the proof of Theorem 3.22, each branch corresponds to a world $\mathcal{D}$ and is marked as either accepting or rejecting depending on the outcome of the logical entailment check $\mathcal{D} \models (Q, \Sigma)$. This logical entailment test is in $\mathfrak{C}$ for the language $\mathcal{L}$, by our assumption. Thus, it can be performed using the oracle. Then, by this construction, the nondeterministic Turing machine answers yes if and only if $\mathrm{P}_{\mathcal{P}}(Q, \Sigma) > p$, which proves membership to $\mathrm{PP}^{\mathfrak{C}}$ in the respective complexity.

To show $\mathfrak{C}$-hardness, we reduce from ontology-mediated query answering, that is, given a database $\mathcal{D}$ and an OMQ (Q, Σ), where Q is a UCQ and Σ is a program over $\mathcal{L}$, decide whether $\mathcal{D} \models (Q, \Sigma)$. We define a PDB $\mathcal{P}$ that contains all the atoms from the database $\mathcal{D}$ with probability 1. Then, it is easy to see that $\mathcal{D} \models (Q, \Sigma)$ if and only if $\mathrm{P}_{\mathcal{P}}(Q) \geq 1$. □

We analyze the consequences of Theorem 7.4. Observe that, for all deterministic complexity classes $\mathfrak{C}$ from Table 6.3 that contain PP, it holds that $\mathrm{PP}^{\mathfrak{C}} = \mathfrak{C}$ and thus Theorem 7.4 directly implies tight complexity bounds. For instance, the data complexity of probabilistic OMQ evaluation for WG is Exp-complete as a simple consequence of Theorem 7.4.

Beyond this generic result, we are interested in lifting the data complexity dichotomy for unions of conjunctive queries to OMQs. Our next result establishes the connection between the respective problems.

Lemma 7.5 *Let (Q, Σ) be an OMQ, where Q is a UCQ, and Σ is a program, and Q_Σ be an FO-rewriting of (Q, Σ). Then, for any PDB $\mathcal{P}$, it holds that $\mathrm{P}_{\mathcal{P}}(Q, \Sigma) = \mathrm{P}_{\mathcal{P}}(Q_\Sigma)$.*

Proof. For any PDB $\mathcal{P}$, it holds that

$$\mathrm{P}_{\mathcal{P}}(Q, \Sigma) \overset{(1)}{=} \sum_{\mathcal{D} \models (Q, \Sigma)} \mathrm{P}_{\mathcal{P}}(\mathcal{D}) \overset{(2)}{=} \sum_{\mathcal{D} \models Q_\Sigma} \mathrm{P}_{\mathcal{P}}(\mathcal{D}) \overset{(3)}{=} \mathrm{P}_{\mathcal{P}}(Q_\Sigma),$$

where (1) follows from Definition 7.1; (2) follows from Q_Σ being the FO-rewriting of Q w.r.t. Σ (see Definition 6.2); and (3) is the definition of the semantics of Q_Σ in PDBs.□

With the help of Lemma 7.5, it becomes possible to lift the data complexity dichotomy in probabilistic databases to all Datalog$^{\pm}$ languages that are FO-rewritable.

Theorem 7.6 (dichotomy) *For all FO-rewritable Datalog$^{\pm}$ languages $\mathcal{L}$, the following holds. $\mathsf{PQE}(\mathsf{UCQ}, \mathcal{L})$ is either in* P *or it is* PP*-complete for PDBs in data complexity under polynomial-time Turing reductions.*

Proof. Let (Q, Σ) be an OMQ, where Q is a UCQ, and Datalog$^{\pm}$ Σ over an FO-rewritable language, and Q_Σ be an FO-rewriting of (Q, Σ). By Lemma 7.5, any polynomial-time algorithm that can evaluate Q_Σ over PDBs also yields the probability of the OMQ (Q, Σ) relative to an PDB, and vice versa. This implies that the OMQ (Q, Σ) is safe if Q_Σ is safe.

Dually, by the same result the probabilities of *all* rewritings of Q coincide, and hence the same algorithm can be used for all of them. Thus, if (Q, Σ) is unsafe, then Q_Σ must also be unsafe for PDBs. By the dichotomy of (Dalvi and Suciu 2012) and Lemma 7.5,

this implies that evaluating the probability of both the UCQ Q_Σ and the OMQ (Q, Σ) must be PP-hard under polynomial-time Turing reductions. □

Obviously, ontological rules introduce dependencies. Therefore, a safe query can become unsafe for OMQs. However, the opposite effect is also possible, i.e., an unsafe query may become safe under ontological rules. We illustrate both of these effects on a synthetic example.

Example 7.7 Consider the conjunctive query $\exists x, y\, \mathsf{C}(x) \wedge \mathsf{D}(x, y)$, which is safe for PDBs. It becomes unsafe under the TGD $\mathsf{R}(x, y), \mathsf{T}(y) \rightarrow \mathsf{D}(x, y)$, since then it rewrites to the query

$$(\exists x, y\, \mathsf{C}(x) \wedge \mathsf{D}(x, y)) \vee (\exists x, y\, \mathsf{C}(x) \wedge \mathsf{R}(x, y) \wedge \mathsf{T}(y)),$$

which is unsafe. Conversely, the conjunctive query $\exists x, y\, \mathsf{C}(x) \wedge \mathsf{R}(x, y) \wedge \mathsf{D}(y)$ is not safe for PDBs, but becomes safe under the TGD $\mathsf{R}(x, y) \rightarrow \mathsf{D}(y)$, as it rewrites to $\exists x, y\, \mathsf{C}(x) \wedge \mathsf{R}(x, y)$. Note that these are very simple TGDs, which are full, acyclic, guarded, and sticky. ◊

Recall that the PP-hardness of probabilistic UCQ evaluation in data complexity holds under polynomial-time Turing reductions (see Chapter 3). This transfers to probabilistic OMQ evaluation relative to FO-rewritable languages. It is open whether probabilistic OMQ evaluation is PP-hard for FO-rewritable languages under polynomial time many-one reductions. On the other hand, for guarded, full programs, we are able to show that PP-hardness applies under such reductions.

Theorem 7.8 $\mathsf{PQE}(\mathsf{UCQ}, \mathsf{GF})$ *is* PP*-hard for PDBs in data complexity.*

Proof. We reduce the following problem (Wagner 1986): decide the validity of the formula $\Phi = \mathsf{C}^c\, x_1, \ldots, x_n\ \varphi$, where $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_k$ is a propositional formula in CNF, over the variables $x_1, \ldots, x_n$. This amounts to checking whether there are at least c assignments to $x_1, \ldots, x_n$ that satisfy φ. We assume without loss of generality that φ contains all clauses of the form $x_j \vee \neg x_j$, $1 \leq j \leq n$; clearly, this does not affect the existence or number of satisfying assignments for φ. For the reduction, we use a PDB $\mathcal{P}_\Phi$ and a program Σ_Φ. We first define the PDB $\mathcal{P}_\Phi$ as follows.

- For each variable x_j, $1 \leq j \leq n$, $\mathcal{P}_\Phi$ contains the probabilistic atoms $\langle \mathsf{L}(x_j, 0) : 0.5 \rangle$ and $\langle \mathsf{L}(x_j, 1) : 0.5 \rangle$, where we view x_j as a constant. These atoms represent the assignments that map x_j to *false* and *true*, respectively.
- For each propositional literal $(\neg)x_\ell$ occurring in a clause φ_j, $1 \leq j \leq k$, $\mathcal{P}_\Phi$ contains the atom $\mathsf{D}(x_\ell, j, i)$ with probability 1, where $i = 1$, if the literal is positive, and $i = 0$, if the literal is negative.
- $\mathcal{P}_\Phi$ contains all the atoms $\mathsf{T}(0)$, $\mathsf{S}(0, 1), \mathsf{S}(1, 2), \ldots, \mathsf{S}(k-1, k)$, $\mathsf{K}(k)$, each with probability 1.

We now describe the program Σ_Φ. To detect when a clause is satisfied, we use the additional unary predicate E and the TGD

$$\mathsf{L}(x, i), \mathsf{D}(y, j, i) \rightarrow \mathsf{E}(j),$$

which is a universally quantified formula over the variables x, y, i and j. We still need to ensure that in each world, exactly one of $\mathsf{L}(x,0)$ and $\mathsf{L}(x,1)$ holds. The clauses $x_j \vee \neg x_j$ take care of the lower bound; for the variables $x_1, \ldots, x_n$, we use the TGDs

$$\mathsf{L}(x,0), \mathsf{L}(x,1) \to \mathsf{B} \qquad \textit{and} \qquad \mathsf{B}, \mathsf{D}(y,j,i) \to \mathsf{E}(j).$$

These TGDs ensure that any inconsistent assignment for $x_1, \ldots, x_n$, i.e., one where some x_j is both *true* and *false*, is automatically marked as satisfying the formula, even if the clause $x_j \vee \neg x_j$ is actually not satisfied. Since there are exactly $4^n - 3^n$ such assignments (where both $\mathsf{L}(x_j,0)$ and $\mathsf{L}(x_j,1)$ hold for at least one x_j), we can add this number to the probability threshold that we will use in the end. Note that the probability of each individual assignment is 0.25^n since there are $2n$ relevant L-atoms (the other atoms are fixed to 0 or 1 and do not contribute here).

It remains to detect whether *all* clauses of φ are satisfied by a consistent assignment, which we do by the means of the TGDs

$$\mathsf{T}(i), \mathsf{S}(i,j), \mathsf{E}(j) \to \mathsf{T}(j) \qquad \textit{and} \qquad \mathsf{T}(i), \mathsf{K}(i) \to \mathsf{Z}(i).$$

Lastly, we define the simple UCQ $Q \coloneqq \exists i\ \mathsf{Z}(i)$. Then, we prove the following claim.

Claim. $\mathrm{P}_{\mathcal{P}_\Phi}(Q, \Sigma_\Phi) \geq 0.25^n(4^n - 3^n + c)$ holds if and only if Φ is valid.

Suppose that Φ is valid, i.e., there are at least c different assignments to $x_1, \ldots, x_n$ that satisfy φ. Then, for each such assignment τ we can define a world $\mathcal{D}_\tau$ such that it contains all atoms from $\mathcal{P}_\Phi$ that occur with probability 1. Moreover, $\mathcal{D}_\tau$ contains an atom $\mathsf{L}(x_j,1)$ if x_j is mapped to true in μ, and an atom $\mathsf{L}(x_j,0)$ if x_j is mapped to false in μ. It is easy to see that each such database $\mathcal{D}_\tau$ is induced by the PDB $\mathcal{P}_\Phi$ and that $\mathcal{D}_t au \models Q$ by our constructions. In particular, this implies that $\mathcal{D}_\tau \models Q_\Phi$ for c worlds. Recall also that $(4^n - 3^n)$ worlds, capturing the inconsistent valuations, satisfy the query. As every world has the probability $(0.5)^{2n}$, we conclude that $\mathrm{P}_{\mathcal{P}_\Phi}(Q_\Phi) \geq 0.5^{2n}(4^n - 3^n + c)$.

Conversely, if the query probability exceeds the threshold value, then some worlds in $\mathcal{P}_\Phi$ with non-zero probability entail (Q, Σ_Φ), i.e., all clauses of φ are satisfied. Each of the non-zero worlds in PDBs represents a unique combination of atoms of the form $\mathsf{L}(x,0)$ and $\mathsf{L}(x,1)$. The worlds where for at least one variable x_j, $1 \leq j \leq n$, neither $\mathsf{L}(x_j,0)$ nor $\mathsf{L}(x_j,1)$ holds do not satisfy φ, and hence do not entail (Q, Σ_Φ) and are not counted. Excluding $(4^n - 3^n)$ worlds, capturing the inconsistent valuations all other worlds represent the actual assignments for $x_1, \ldots, x_n$, and hence we know that at least c of those satisfy φ. Thus, we conclude that Φ is valid.

Observe that all TGDs used in the reduction are *full* and *guarded*. Moreover, only the PDB and the probability threshold depend on the input formula (which is allowed in data complexity). Hence, the reduction shows PP-hardness of $\mathsf{PQE}(\mathsf{GF}, \mathsf{UCQ})$ for PDBs in data complexity. □

As pointed out before, GF is one of the least expressive Datalog$^{\pm}$ languages with polynomial time data complexity for OMQA. Thus, this result already implies PP-hardness for the classes G, F, WS and WA. This completes all results regarding the data complexity.

All of the upper bounds for the fixed-program complexity are a consequence of Theorem 7.4. For instance, probabilistic OMQ evaluation for WG is in $\mathrm{PP}^{\mathrm{NP}}$ since

OMQA for WA is NP-complete. It only remains to show PP^{NP}-hardness for these languages, which already follows from Theorem 3.23. Recall that probabilistic query evaluation for unions of conjunctive queries is already PP^{NP}-hard in combined complexity by this result. Since the query is not fixed in fixed-program complexity, this hardness also applies to OMQs in fixed-program complexity (even if the program is empty) and clearly also to the bounded-arity complexity. This completes the picture for fixed-program complexity results.

Analogous arguments yield tight complexity bounds also for the bounded-arity and combined complexity with the only exception being the class A. More generally, Theorem 7.4 does not yield tight complexity bounds for languages where OMQA is NEXP-complete as it is not known whether $\text{PP}^{\text{NEXP}} \subseteq \text{NEXP}$. On the other hand, we observe that the non-determinism in the oracle NEXP calls are used in a restricted fashion, which allows us to solve the problem in NEXP, as we show next.

Theorem 7.9 $\mathsf{PQE}(\mathsf{UCQ}, \mathcal{L})$ *is* NEXP*-complete for PDBs in data (respectively, bounded-arity, fixed-program, combined) complexity if ontology-mediated query answering in* $\mathcal{L}$ *is* NEXP*-complete in data (respectively, bounded-arity, fixed-program, combined) complexity.*

Proof. Let (Q, Σ) be an OMQ, $\mathcal{P}$ a PDB and p a threshold value. Consider an exponential time nondeterministic Turing machine which enumerates all worlds $\mathcal{D}$ (of which there are exponentially many) and adds up their probability $\text{P}_{\mathcal{P}}(\mathcal{D})$ if the test $\mathcal{D} \models (Q, \Sigma)$ is successful. Observe that this means exponentially many NEXP tests. Then, the nondeterministic Turing machine answers yes if and only if $\text{P}_{\mathcal{P}}(Q, \Sigma) > p$. □

By this result, we conclude our complexity analysis. We give an overview of the results in relation to the results obtained in Chapter 3.

7.1.2 Overview of the Results

The complexity results for probabilistic OMQ evaluation is summarized in Table 7.2. We lifted the data complexity dichotomy in PDBs for unions of conjunctive queries to OMQs relative to FO-rewritable languages: by this result an OMQ can be evaluated either in P or it is PP-hard. We note that a similar result was obtained earlier for the DL *DL-Lite* in (Jung and Lutz 2012).

All the results except the dichotomy result are obtained under standard many-one reductions. Interestingly, for classes where OMQA is P-hard in data complexity, we are able to show PP-hardness under polynomial time many-one reductions. However, it is open whether the data complexity dichotomy can be extended to these languages. Note that these languages are Datalog-rewritable, meaning that a data complexity dichotomy in these languages is essentially related to a similar result in Datalog.

For the bounded-arity, fixed-program and combined complexity, we observe that the complexity of OMQA is dominating the complexity of probabilistic OMQ evaluation in all cases except for the PP^{NP}-completeness results, for which hardness follows from Theorem 3.23. This concludes our analysis for ontology-mediated queries for PDBs.

Table 7.2: Complexity of probabilistic OMQ evaluation for PDBs. All first-order rewritable languages admit a data complexity dichotomy between P and PP under poynomial time Turing reductions.

Datalog$^{\pm}$ Languages	data	fixed-program	bounded-arity	combined
L, LF, AF	P vs PP [Theorem 7.6]	PP^{NP} [Theorem 3.23, 7.4]	PP^{NP} [Theorem 3.23, 7.4]	PSpace [Theorem 7.4]
S, SF	P vs PP [Theorem 7.6]	PP^{NP} [Theorem 3.23, 7.4]	PP^{NP} [Theorem 3.23, 7.4]	Exp [Theorem 7.4]
A	P vs PP [Theorem 7.6]	PP^{NP} [Theorem 3.23, 7.4]	NExp [Theorem 7.9]	NExp [Theorem 7.9]
GF, F	PP [Theorem 7.4, 7.8,]	PP^{NP} [Theorem 3.23, 7.4]	PP^{NP} [Theorem 3.23]	Exp [Theorem 7.4]
G	PP [Theorem 7.4, 7.8]	PP^{NP} [Theorem 3.23, 7.4]	Exp [Theorem 7.4]	2Exp [Theorem 7.4]
WS, WA	PP [Theorem 7.4, 7.8]	PP^{NP} [Theorem 3.23]	2Exp [Theorem 7.4]	2Exp [Theorem 7.4]
WG	Exp [Theorem 7.4]	Exp [Theorem 7.4]	Exp [Theorem 7.4]	2Exp [Theorem 7.4]

7.2 Ontology-Mediated Queries for Open-World Probabilistic Databases

We have introduced OpenPDBs, which generalize PDBs to be able to deal with incompleteness. More precisely, in OpenPDBs the probabilities of facts that are not in the database, called open atoms, are relaxed to a default probability interval, which is very different from the CWA of PDBs, which requires the probabilities of such facts to be zero. In the resulting framework of OpenPDBs, query probabilities are given in terms of upper and lower probability values, which is more in line with an incomplete view of the world.

While forming a natural and flexible basis for querying incomplete data sources, OpenPDBs are limited in the following sense: All open atoms can take on probability values from a single *fixed* interval $[0, \lambda]$, which results in the *same* upper and lower probabilities for many queries. We illustrate this by extending Example 7.2.

Example 7.10 Let us consider the OpenPDB $\mathcal{G}_a = (\mathcal{P}_a, 0.5)$ where the PDB $\mathcal{P}_a$ is given in Table 7.1. In OpenPDBs, Writer(hamsun) and Writer(goriot) evaluate to the *same lower and upper probabilities* (0 and 0.5, respectively), since both atoms are open. Intuition, however, tells us that hamsun is more likely to be a writer, as we already know from the given PDB (with high confidence) that hamsun has authored a novel. On the other hand, Writer(goriot) is unlikely to hold, since we know (with high confidence) that goriot is a novel. Essentially, we lack the common-sense knowledge that

(i) anyone who has authored a novel is a writer, and

(ii) writers and novels are disjoint entities,

which helps us to distinguish such queries. Observe that (i) is a positive axiom and would lead to higher probabilities, whereas (ii) is a negative (constraining) axiom and would entail lower probabilities for some queries. More precisely, we are talking about the rules

$$\forall x, y\, \mathsf{AuthorOf}(x, y) \wedge \mathsf{Novel}(y) \rightarrow \mathsf{Writer}(x),$$
$$\forall x\, \mathsf{Writer}(x) \wedge \mathsf{Novel}(x) \rightarrow \bot,$$

where the first axiom is the TGD (6.3) and the second axiom the NC (6.1). ◊

In essence, many atoms from the open-world evaluate to the same default probability and this results in highly symmetric probabilities for queries. Thus, it is crucial to restrict the open-world to provide tighter, and thus more informative, probability bounds. Our proposal is based on ontological rules which, intuitively, break down such symmetries and allow us to distinguish queries which were indistinguishable before, as in Example 7.10.

The semantics is again based on maximizing (respectively, minimizing) over a credal set of probability distributions. The difference is that we restrict our attention to consistent distributions and we assume that the input PDB is consistent.

Definition 7.11 (semantics) The *probability interval of an OMQ* (Q, Σ) *for an Open-PDB* $\mathcal{G} = (\mathcal{P}, \lambda)$ is given by $\mathrm{K}_{\mathcal{G}}(Q, \Sigma) \coloneqq [\underline{\mathrm{P}}_{\mathcal{G}}(Q, \Sigma), \overline{\mathrm{P}}_{\mathcal{G}}(Q, \Sigma)]$, where

$$\underline{\mathrm{P}}_{\mathcal{G}}(Q, \Sigma) \coloneqq \min_{\mathrm{P} \in \mathrm{K}_{\mathcal{G}}} \{\mathrm{P}(Q, \Sigma) \mid \mathrm{P} \text{ is consistent w.r.t. } \Sigma\},$$
$$\overline{\mathrm{P}}_{\mathcal{G}}(Q, \Sigma) \coloneqq \max_{\mathrm{P} \in \mathrm{K}_{\mathcal{G}}} \{\mathrm{P}(Q, \Sigma) \mid \mathrm{P} \text{ is consistent w.r.t. } \Sigma\}.$$

The special case of $\lambda = 0$ corresponds to having a single (closed-world) PDB $\mathcal{P}$. In this case, we simply speak of the *probability of* (Q, Σ) *for a PDB* $\mathcal{P}$. ◊

In the following, we evaluate this semantics with respect to the goals identified in the motivation, and discuss our choice of restricting to the consistent λ-completions.

We argued that OpenPDBs can benefit from an axiomatic encoding of the knowledge of the domain. Consider again our running example, which is now enriched with a program. How do the queries evaluate under this semantics?

Example 7.12 Consider again the Example 7.10 with the given query semantics. Observe that Writer(hamsun) and Writer(goriot) do not evaluate to the *same* lower and upper probabilities any more, although both atoms are open.

The lower probability of Writer(goriot) remains 0, while the upper probability now decreases to 0.15 from 0.5 as a consequence of the NC (6.1). In contrast, the lower probability of Writer(hamsun) increases to 0.63, while the upper probability increases to 0.815 due to the TGD (6.3). These intervals are much more informative than the default interval $[0, 0.5]$. ◊

The most subtle aspect of choosing the *best* distribution is the question of how to deal with inconsistent worlds. Ignoring inconsistencies (and optimizing over *all* completions) leads to a drowning effect: since inconsistent worlds entail everything, this semantics would be biased towards choosing inconsistent λ-completions. This does not satisfy our goals, as even an unsatisfiable query could evaluate to a positive probability.

An alternative approach, which is standard for (closed-world) PDBs, and is quite intuitive at first glance, would be to choose the distribution which maximizes the probability

$$(\mathrm{P}(Q, \Sigma) - \gamma)/(1 - \gamma),$$

where γ is the normalization factor (as defined before). A careful inspection, however, shows that this semantics also favors inconsistent distributions over consistent ones.

This is mainly due to the normalization process internal to the computation. As part of this normalization, the probability mass of inconsistent worlds is distributed to consistent worlds (as explained before). As a consequence, it is often possible to increase the query probability by simply increasing the probability of inconsistent worlds. This is not a desired effect, since we are interested in finding the most suitable λ-completion from the open-world, and not the one that increases the query probability by increasing the probability mass of inconsistent worlds.

To avoid such drowning effects, our proposal considers only consistent distributions. That is, we do not want to introduce inconsistencies when completing our knowledge over the domain by choosing a λ-completion. One drawback of our approach is the fact that inconsistencies are not tolerated even if the inconsistency degree is very small. However, it would be easy to introduce a threshold value, say 0.1, to tolerate the inconsistent completions where the probability of the inconsistent worlds does not exceed this threshold.

We discuss other semantic possibilities and future directions in the end of this chapter. We now reformulate the problem of lower and upper probabilistic query evaluation for OMQs.

Definition 7.13 (lower, upper probabilistic query evaluation) Given a PDB $\mathcal{P}$, an OMQ (Q, Σ) and a value $p \in [0, 1]$; *upper probabilistic query evaluation*, denoted $\overline{\mathsf{P}}\mathsf{QE}$, is to decide whether $\overline{\mathrm{P}}_{\mathcal{G}}(Q, \Sigma) > p$ holds and *lower probabilistic query evaluation*, denoted $\overline{\mathsf{P}}\mathsf{QE}$, is to decide whether $\underline{\mathrm{P}}_{\mathcal{G}}(Q, \Sigma) > p$ holds. Both $\overline{\mathsf{P}}\mathsf{QE}$ and $\underline{\mathsf{P}}\mathsf{QE}$ are parametrized with the language of the ontology-mediated query; we write $\overline{\mathsf{P}}\mathsf{QE}(\mathsf{Q}, \mathcal{L})$ (respectively, $\underline{\mathsf{P}}\mathsf{QE}(\mathsf{Q}, \mathcal{L})$) to refer to $\overline{\mathsf{P}}\mathsf{QE}$ (respectively, $\underline{\mathsf{P}}\mathsf{QE}$) on the class Q of queries and on the class of ontologies restricted to the language $\mathcal{L}$. $\Diamond$

7.2.1 Complexity Results

We restrict our complexity analysis always to the bounded-arity complexity since allowing arbitrary arity immediately results in high complexity classes that are beyond the scope of this thesis. More precisely, we consider data complexity and bounded-arity complexity as before and omit combined complexity. Moreover, in fixed-program complexity, we additional assume a bounded-arity schema and to avoid ambiguity, we call it the *fixed-program&arity complexity*.

In the following, we separate the complexity analysis into two parts; first, we investigate the special case of positive programs and then extend our treatment to programs that also contain negative constraints.

Positive Programs. We first show an analogous result to Lemma 7.5 for positive programs.

Lemma 7.14 *Let (Q, Σ^+) be an OMQ, where Q is a UCQ, and Σ^+ is a positive Datalog$^\pm$ program, and Q_{Σ^+} be an FO-rewriting of (Q, Σ^+). Then, for any OpenPDB $\mathcal{G}$, it holds that $\overline{\mathrm{P}}_{\mathcal{G}}(Q, \Sigma^+) = \overline{\mathrm{P}}_{\mathcal{G}}(Q_{\Sigma^+})$ and $\underline{\mathrm{P}}_{\mathcal{G}}(Q, \Sigma^+) = \underline{\mathrm{P}}_{\mathcal{G}}(Q_{\Sigma^+})$.*

Proof. For any OpenPDB $\mathcal{G}$, it holds that

$$\overline{\mathrm{P}}_{\mathcal{G}}(Q, \Sigma^+) \overset{(1)}{=} \max\{\mathrm{P}(Q, \Sigma^+) \mid \mathrm{P} \in \mathrm{K}_{\mathcal{G}}\} \overset{(2)}{=} \max\{\mathrm{P}(Q_{\Sigma^+}) \mid \mathrm{P} \in \mathrm{K}_{\mathcal{G}}\} \overset{(3)}{=} \overline{\mathrm{P}}_{\mathcal{G}}(Q_{\Sigma^+}),$$
$$\underline{\mathrm{P}}_{\mathcal{G}}(Q, \Sigma^+) \overset{(1)}{=} \min\{\mathrm{P}(Q, \Sigma^+) \mid \mathrm{P} \in \mathrm{K}_{\mathcal{G}}\} \overset{(2)}{=} \min\{\mathrm{P}(Q_{\Sigma^+}) \mid \mathrm{P} \in \mathrm{K}_{\mathcal{G}}\} \overset{(3)}{=} \underline{\mathrm{P}}_{\mathcal{G}}(Q_{\Sigma^+}).$$

where (1) is the special case of the semantics given in Definition 7.11 that applies to positive programs; (2) follows from Q_Σ being the FO-rewriting of Q w.r.t. Σ; and (3) follows from the semantics of Q_Σ in OpenPDBs. □

Notice that the following is then an immediate consequence of Lemma 7.14, Lemma 7.5 and Theorem 4.17:

$$\mathrm{K}_{\mathcal{G}}(Q, \Sigma^+) = [\mathrm{P}_{\mathcal{P}_0}(Q, \Sigma^+), \mathrm{P}_{\mathcal{P}_\lambda}(Q, \Sigma^+)],$$

since the query Q is a UCQ and Σ^+ is a positive FO-rewritable program. More importantly, with the help of Lemma 7.14, we can lift the data complexity dichotomy of OpenPDBs to all FO-rewritable, positive programs in a straightforward manner.

Theorem 7.15 (dichotomy) *For all positive FO-rewritable Datalog$^\pm$ languages $\mathcal{L}^+$, $\overline{\mathsf{PQE}}(\mathcal{L}^+, \mathsf{UCQ})$ and $\underline{\mathsf{PQE}}(\mathcal{L}^+, \mathsf{UCQ})$ are either in* P *or* PP*-complete for OpenPDBs in data complexity under polynomial-time Turing reductions.*

Proof. Let (Q, Σ^+) be an OMQ, where Q is a UCQ, Σ^+ a positive program over an FO-rewritable language, and Q_{Σ^+} be an FO-rewriting of (Q, Σ^+). By Lemma 7.14, any polynomial time algorithm that can evaluate Q_{Σ^+} over OpenPDBs also yields the probability of the OMQ (Q, Σ^+) relative to a OpenPDB, and vice versa. This implies that the OMQ (Q, Σ^+) is safe if Q_{Σ^+} is safe.

Dually, by the same result the probabilities of *all* rewritings of Q coincide, and hence the same algorithm can be used for all of them. Thus, if (Q, Σ^+) is unsafe, then Q_{Σ^+} must also be unsafe for OpenPDBs. By the data complexity dichotomy in OpenPDBs, given in Theorem 4.24, this implies that evaluating the probability of both the UCQ Q_{Σ^+} and the OMQ (Q, Σ^+) must be PP-hard. □

Note that Lemma 7.14 is based on the fact that programs are positive, that is, choosing a consistent completion that maximizes (respectively, minimizes) the query probability can be done in an efficient manner, which helps us to obtain the data complexity dichotomy. This picture changes if we go beyond positive programs.

Beyond Positive Programs. In the presence of negative constraints, it still suffices to consider the extremal λ-completions. In fact, once the correct completion is known, the probabilistic OMQ evaluation for OpenPDBs can still be reduced to probabilistic OMQ evaluation for PDBs. The key difference in the presence of NCs is that we have to make sure that this completion is consistent. That is, choosing the completion $\mathcal{P}_\lambda$ that sets all open tuples to λ is not feasible, as this will very likely lead to inconsistencies. However, observe that the *lower* probability can still be obtained from the completion $\mathcal{P}_0$

(which we assumed to be consistent), and hence the previous results still hold for lower probabilistic OMQ evaluation with NCs.

A naïve way of solving the upper probabilistic OMQ evaluation is to *guess* a λ-completion and then check whether it is consistent and compare the resulting probability to the threshold.

Theorem 7.16 *Let $\mathfrak{C}$ denote the data (respectively, fixed-program&arity, bounded-arity) complexity of ontology-mediated query answering for a Datalog$^{\pm}$ language $\mathcal{L}$. Then, $\overline{\mathsf{PQE}}(\mathsf{UCQ}, \mathcal{L})$ is $\mathfrak{C}$-hard and in $\mathrm{NP}^{\mathrm{PP}^{\mathfrak{C}}}$ for OpenPDBs in data (respectively, fixed-program&arity, bounded-arity) complexity.*

Proof. Let (Q, Σ) be an OMQ, $\mathcal{G} = (\mathcal{P}, \lambda)$ be an OpenPDB, and p be a threshold value. We consider a nondeterministic Turing machine with an oracle access to $\mathrm{PP}^{\mathfrak{C}}$. To decide $\overline{\mathrm{P}}_{\mathcal{G}}(Q, \Sigma)$, we can guess a λ-completion $\hat{\mathcal{P}}$ and decide whether $\mathrm{P}_{\hat{\mathcal{P}}}(Q, \Sigma) > p$ using the $\mathrm{PP}^{\mathfrak{C}}$ oracle, which is possible due to Theorem 7.4. Analogously, to decide $\underline{\mathrm{P}}_{\mathcal{G}}(Q, \Sigma)$, we can verify whether for all λ-completions $\hat{\mathcal{P}}$ it holds that $\mathrm{P}_{\hat{\mathcal{P}}}(Q, \Sigma) > p$. $\mathfrak{C}$-hardness is an immediate consequence of Theorem 7.4. □

The main question is, of course, whether we really need to guess a completion, or can it still be computed in an efficient manner? We provide a partial answer to this question: for guarded, full programs, we are able to show $\mathrm{NP}^{\mathrm{PP}}$-hardness in data complexity. In more intuitive terms, this implies that probabilistic OMQ evaluation is harder for OpenPDBs than it is for PDBs in these languages.

Theorem 7.17 $\overline{\mathsf{PQE}}(\mathsf{UCQ}, \mathsf{GF})$ *is $\mathrm{NP}^{\mathrm{PP}}$-hard for OpenPDBs in data complexity.*

Proof. We reduce the following $\mathrm{NP}^{\mathrm{PP}}$-complete problem (Wagner 1986): decide the validity of the formula

$$\Phi = \exists x_1, \ldots, x_\ell \, \mathsf{C}^c \, y_1, \ldots, y_m \; \varphi,$$

where $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_k$ is a propositional formula in CNF, over the variables $x_1, \ldots, x_\ell$, $y_1, \ldots, y_m$. This amounts to checking whether there is a partial assignment for $x_1, \ldots, x_\ell$ that admits at least c extensions to $y_1, \ldots, y_m$ that satisfy φ.

As before, we assume without loss of generality that φ contains all clauses of the form $x_j \vee \neg x_j$, $1 \leq j \leq \ell$, and similarly $y_j \vee \neg y_j$, $1 \leq j \leq m$; clearly, this does not affect the existence or number of satisfying assignments for φ.

We define the PDB $\mathcal{P}$.

- For each variable y_j, $1 \leq j \leq m$, it contains the tuples $\langle \mathsf{L}(y_j, 0) : 0.5 \rangle$ and $\langle \mathsf{L}(y_j, 1) : 0.5 \rangle$, where we view y_j as a constant. These tuple represent the assignments that map y_j to *false* and *true*, respectively.
- For each literal $(\neg)x$ occurring in a clause φ_j, $1 \leq j \leq k$, we add the tuple $\mathsf{D}(x, j, i)$ with probability 1, where $i = 1$, if the literal is positive, and $i = 0$, if the literal is negative.
- We add the tuples $\mathsf{T}(0), \mathsf{S}(0, 1), \mathsf{S}(1, 2), \ldots, \mathsf{S}(k-1, k), \mathsf{K}(k)$, each with probability 1.

Moreover, for each variable x_j, $1 \leq j \leq \ell$, we need two open atoms $\mathsf{P}(x_j, 0)$ and $\mathsf{P}(x_j, 1)$ with similar semantics as the L-atoms, and we set $\lambda := 1$. All other atoms over the introduced signature are added to $\mathcal{P}_\Phi$ with probability 0.

We now describe the program Σ. To detect when a clause is satisfied, we use the additional unary predicate E and the TGDs

$$\mathsf{P}(x,i), \mathsf{D}(x,j,i) \to \mathsf{E}(j) \quad \text{and} \quad \mathsf{L}(y,i), \mathsf{D}(y,j,i) \to \mathsf{E}(j).$$

However, we still need to ensure that in each world, exactly one of $\mathsf{P}(x,0)$ and $\mathsf{P}(x,1)$ holds, and similarly for L. The clauses $x_j \vee \neg x_j$ and $y_j \vee \neg y_j$ take care of the lower bound; for the variables $x_1, \ldots, x_\ell$ we can represent the remaining part of this constraint through the NC

$$\mathsf{P}(x,0), \mathsf{P}(x,1) \to \bot.$$

This ensures that each consistent λ-completion (that satisfies φ in an as yet unspecified way) represents exactly one truth assignment for the variables $x_1, \ldots, x_\ell$; moreover, every such assignment can be expressed as a consistent λ-completion. For the variables $y_1, \ldots, y_m$, a similar NC would yield only inconsistent completions. Instead, we use the TGDs

$$\mathsf{L}(y,0), \mathsf{L}(y,1) \to \mathsf{B} \quad \text{and} \quad \mathsf{B}, \mathsf{D}(x,j,i) \to \mathsf{E}(j).$$

These ensure that any inconsistent assignment for $y_1, \ldots, y_m$, i.e., one where some y_j is both *true* and *false*, is automatically marked as satisfying the formula, even if the clauses $x_j \vee \neg x_j$ and $y_j \vee \neg y_j$ are not actually satisfied. Since there are exactly $4^m - 3^m$ such assignments (where both $\mathsf{L}(y_j,0)$ and $\mathsf{L}(y_j,1)$ hold for at least one y_j), we can add this number to the probability threshold that we will use in the end. Note that the probability of each individual assignment is 0.25^m since there are $2m$ relevant L-tuples (the other tuples are fixed to 0 or 1 and do not contribute here).

It remains to detect whether *all* clauses of φ are satisfied by a consistent assignment, which we do by the means of the TGDs

$$\mathsf{T}(i), \mathsf{S}(i,j), \mathsf{E}(j) \to \mathsf{T}(j) \quad \text{and} \quad \mathsf{T}(i), \mathsf{K}(i) \to \mathsf{Z}(i)$$

and, finally, we consider the simple conjunctive query $Q \coloneqq \exists i\ \mathsf{Z}(i)$. We define the OpenPDB $\mathcal{G} = (\mathcal{P}, 1)$, where $\mathcal{P}$ is defined as above and for the OMQ (Q, Σ) prove the following claim.

Claim. $\overline{\mathrm{P}}_{\mathcal{G}}(Q,\Sigma) > 0.25^m(4^m - 3^m + (c-1))$ holds if and only if Φ is satisfiable.

If $\overline{\mathrm{P}}_{\mathcal{G}}(Q,\Sigma) > 0.25^m(4^m - 3^m + (c-1))$, then there is a λ-completion in which the query probability exceeds this value, which means that at least some worlds with non-zero probability entail (Q,Σ), i.e., all clauses of φ are satisfied. Hence, this λ-completion represents a valid assignment of the variables $x_1, \ldots, x_\ell$. Each of the non-zero worlds under this completion represents a unique combination of atoms of the form $\mathsf{L}(y,0)$ and $\mathsf{L}(y,1)$. The worlds where for at least one variable y_j, $1 \leq j \leq m$, neither $\mathsf{L}(y_j,0)$ nor $\mathsf{L}(y_j,1)$ holds do not satisfy φ, and hence do not entail (Q,Σ) and are not counted. Of the remaining worlds, $4^m - 3^m$ automatically entail (Q,Σ). The other worlds represent the actual assignments for $y_1, \ldots, y_m$, and hence we know that more than $c-1$ of those satisfy φ.

Conversely, if we are given a partial assignment for $x_1, \ldots, x_\ell$ that satisfies this property, then it is easy to construct a λ-completion as above and show that it exceeds the given threshold, using the ideas described above.

All TGDs used here are *full* and *guarded*. Moreover, only the PDB and the probability threshold depend on the input formula. Hence, the reduction shows $\mathrm{NP}^{\mathrm{PP}}$-hardness of upper probabilistic query evaluation in GF. □

What remains open is whether a similar hardness result can be obtained for FO-rewritable languages. Notice that, even if such a result holds, it has to differ significantly from the techniques employed in the proof of Theorem 7.17, which deliberately make use of guarded TGDs in the reduction.

Nevertheless, if we compute the complexity while taking also the query into account, we can show a similar $\mathrm{NP}^{\mathrm{PP}}$-hardness result for FO-rewritable languages. Importantly, this result is very different from the previous one as it uses the expressive power of the query (instead of the TGDs) and the program contains only NCs.

Theorem 7.18 *Given a Datalog$^{\pm}$ language $\mathcal{L}$ where the fixed-program&arity (respectively, bounded-arity) complexity of ontology-mediated query answering is* NP*-complete, it holds that $\overline{\mathsf{PQE}}(\mathsf{UCQ}, \mathcal{L})$ is $\mathrm{NP}^{\mathrm{PP}}$-complete for OpenPDBs in fixed-program&arity (respectively, bounded-arity) complexity.*

Proof. The upper bound already follows from Theorem 7.16. Thus, we only show $\mathrm{NP}^{\mathrm{PP}}$-hardness, for which we reduce the following problem: decide validity of

$$\Phi = \exists x_1, \ldots, x_\ell \, \mathsf{C}^c \, y_1, \ldots, y_m \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k \,,$$

where every φ_i is a propositional clause over $x_1, \ldots, x_\ell$, $y_1, \ldots, y_m$, and $k, \ell, m \geq 1$. As in the proof of Theorem 7.17, we can assume without loss of generality that φ contains all clauses of the form $x_j \vee \neg x_j$, $1 \leq j \leq \ell$, and $y_j \vee \neg y_j$, $1 \leq j \leq m$. We will also assume that each clause φ_j contains exactly three literals.

The PDB $\mathcal{P}_\Phi$ for the reduction is defined as follows.

- For each variable y_j, $1 \leq j \leq m$, it contains the atoms $\langle \mathsf{L}(y_j, 0) : 0.5 \rangle$ and $\langle \mathsf{L}(y_j, 1) : 0.5 \rangle$.
- Each clause φ_j is described with the help of a predicate $\mathsf{M}(\cdot, \cdot, \cdot, j)$ of arity 4, which encodes the satisfying assignments for φ_j. For example, consider the clause $\varphi_j = x_2 \vee \neg y_4 \vee y_1$. For the satisfying assignment $x_2 \mapsto \mathit{true}$, $y_4 \mapsto \mathit{true}$, $y_1 \mapsto \mathit{false}$, we add the tuple $\mathsf{M}(1, 1, 0, j)$ with probability 1, and similarly for all other satisfying assignments. There are at most 7 satisfying assignments for each clause.

We use the open atoms $\mathsf{P}(x_j, 0)$ and $\mathsf{P}(x_j, 1)$ for the variables x_j, $1 \leq j \leq \ell$, set $\lambda \coloneqq 1$, and fix all other possible atoms to the probability 0. We define the program Σ_Φ for the reduction as follows. We use the NC $\mathsf{P}(x, 0), \mathsf{P}(x, 1) \rightarrow \bot$ to enforce that the variables x_j, $1 \leq j \leq \ell$, get a correct truth assignment. However, we do not employ any TGDs. The UCQ for which we will check entailment is

$$Q_\Phi \coloneqq (\psi_1 \wedge \cdots \wedge \psi_k) \vee (\exists y \, \mathsf{L}(y, 0) \wedge \mathsf{L}(y, 1)),$$

where each ψ_j is a conjunction that is derived from φ_j depending on the types of the involved variables. We describe the details again on the example clause $\varphi_j = x_2 \vee \neg y_4 \vee y_1$. The satisfaction of this clause is encoded by the conjunction

$$\psi_j = \mathsf{M}(i_1, i_2, i_3, j) \wedge \mathsf{P}(x_2, i_1) \wedge \mathsf{L}(y_4, i_2) \wedge \mathsf{L}(y_1, i_3),$$

where i_1, i_2, i_3 are existentially quantified variables that are local to ψ_j, and j is fixed. Intuitively, ψ_j asserts that the truth assignment for x_2, y_4, and y_1 (given by i_1, i_2, and i_3, respectively) satisfies φ_j. The assignment for the variables $x_1, \ldots, x_\ell$, $y_1, \ldots, y_m$ is fixed by the current λ-completion (using P) and world (using L), respectively.

An alternative way of satisfying Q_Φ is that L represents an inconsistent assignment for at least one variable of the form y_j, which again happens in exactly $4^m - 3^m$ worlds. Thus, given the OpenPDB $\mathcal{G} = (\mathcal{P}_\Phi, 1)$, it remains to prove the following claim.

Claim. $\overline{\mathrm{P}}_\mathcal{G}(Q_\Phi, \Sigma_\Phi) > 0.25^m(4^m - 3^m + (c-1))$ holds if and only if Φ is valid.

If $\overline{\mathrm{P}}_\mathcal{G}(Q_\Phi, \Sigma_\Phi) > 0.25^m(4^m - 3^m + (c-1))$, then there exists at least one λ-completion that obtains this value. This λ-completion must represent a valid assignment for the variables $x_1, \ldots, x_\ell$ since otherwise only $4^m - 3^m$ worlds satisfy (Q_Φ, Σ_Φ). Of the 3^m worlds that do not satisfy $\exists y\, \mathsf{L}(y, 0) \wedge \mathsf{L}(y, 1)$ there are at most 2^m that also satisfy the constraints on the variables $y_1, \ldots, y_m$, and hence represent a valid extension to an assignment for $y_1, \ldots, y_m$. Thus, there must be at least c assignments for $y_1, \ldots, y_m$ that extend the partial assignment, which means that Φ is valid.

Conversely, if Φ is valid, then there exists an assignment for $x_1, \ldots, x_\ell$ (which induces a λ-completion), for which there are at least c extensions to $y_1, \ldots, y_m$ (and hence at least $4^m - 3^m + c$ worlds) that satisfy $\varphi_1, \ldots, \varphi_k$ (and hence (Q_Φ, Σ_Φ) is satisfied). This shows that $\overline{\mathrm{P}}_\mathcal{G}(Q_\Phi, \Sigma_\Phi)$ exceeds the given threshold. Since the reduction is w.r.t. a bounded arity, we did not use any TGDs and the only NC that was used does not depend on Φ, this shows the claim. □

Notice that Theorem 7.16 together with Theorem 7.17 and Theorem 7.18 yields tight complexity bounds for all the cases in fixed-program&arity and bounded-arity complexity except the class A. We obtain an analogous result for this case as in Theorem 7.9.

Theorem 7.19 $\overline{\mathsf{PQE}}(\mathsf{UCQ}, \mathcal{L})$ *is* NExp*-complete for OpenPDBs in bounded-arity complexity if ontology-mediated query answering in* $\mathcal{L}$ *is* NExp*-complete in the bounded-arity complexity.*

Proof. Let (Q, Σ) be an OMQ, where Σ is over $\mathcal{L}$, $\mathcal{G} = (\mathcal{P}, \lambda)$ be an OpenPDB, and p be a threshold value. Hardness is a consequence of Theorem 7.16. We only need to prove membership. By Theorem 7.9, the probabilistic OMQ evaluation problem $\mathrm{P}_{\mathcal{P}'}(Q, \Sigma) > p$ for PDBs is NExp-complete in bounded-arity complexity. Thus, to decide the *upper* probabilistic query evaluation, we can go through all λ-completions (of which there are exponentially many in bounded-arity complexity) and verify in NExp whether there is a completion $\hat{\mathcal{P}}$ for which it holds that $\mathrm{P}_{\hat{\mathcal{P}}}(Q, \Sigma) > p$. □

This concludes our complexity analysis. Notice that we have excluded the combined complexity from our analysis and assumed a bounded-arity for the fixed-program complexity.

7.2.2 Overview of the Results

The complexity results for upper probabilistic OMQ evaluation are summarized in Table 7.3. As already pointed out, the complexity of lower probabilistic OMQ evaluation for OpenPDBs coincides with probabilistic OMQ evaluation for PDBs.

Table 7.3: Complexity of upper probabilistic OMQ evaluation for OpenPDBs. Combined complexity is excluded from the analysis. Lower probabilistic OMQ evaluation for OpenPDBs coincides with the case of PDBs; see Table 7.2.

Datalog$^{\pm}$ Languages	data	fixed-program&arity	bounded-arity
L, LF, AF, S SF	$\leq$ NP$^{\text{PP}}$ [Theorem 7.16]	NP$^{\text{PP}}$ [Theorem 7.18]	NP$^{\text{PP}}$ [Theorem 7.18]
A	$\leq$ NP$^{\text{PP}}$ [Theorem 7.16]	NP$^{\text{PP}}$ [Theorem 7.16]	NExp [Theorem 7.19]
GF F	NP$^{\text{PP}}$ [Theorem 7.17]	NP$^{\text{PP}}$ [Theorem 7.17]	NP$^{\text{PP}}$ [Theorem 7.18]
G	NP$^{\text{PP}}$ [Theorem 7.17]	NP$^{\text{PP}}$ [Theorem 7.17]	Exp [Theorem 7.16]
WS, WA	NP$^{\text{PP}}$ [Theorem 7.17]	2Exp [Theorem 7.16]	2Exp [Theorem 7.16]
WG	Exp [Theorem 7.16]	Exp [Theorem 7.16]	Exp [Theorem 7.16]

We lifted the data complexity dichotomy in OpenPDBs for unions of conjunctive queries to probabilistic OMQ evaluation for FO-rewritable languages. Importantly, this applies only to positive programs and it remains open whether a similar dichotomy can be obtained in the presence of NCs (which is also closely connected to the quest of obtaining a dichotomy for nonmonotone queries in PDBs).

As before, all the results except the dichotomy result are obtained under standard many-one reductions. In a significant result for classes, where OMQA is P-hard in data complexity, we are able to show NP$^{\text{PP}}$-hardness for upper probabilistic OMQ evaluation for OpenPDBs. This result highly contrasts with the PP-completeness result in data complexity for PDBs. It is open whether a similar hardness result applies to FO-rewritable languages in data complexity.

Unsurprisingly, probabilistic query evaluation in OpenPDBs is typically harder than probabilistic query evaluation in PDBs. For the fixed-program&arity and bounded-arity complexity, we also show NP$^{\text{PP}}$-completeness for upper probabilistic OMQ evaluation, which contrasts with the PP$^{\text{NP}}$-completeness results from Table 7.2. Recall that PP$^{\text{NP}} \subseteq$ NP$^{\text{PP}}$. As before, we observe that the complexity of OMQA is dominating the complexity of probabilistic OMQ evaluation in all other cases.

7.3 Most Probable Explanations for Ontology-Mediated Queries

Motivated by maximal posterior computations in PGMs, we studied the most probable database and most probable hypothesis problems in PDBs (see Chapter 5). We now extend our results towards ontology-mediated queries. In a nutshell, we restrict ourselves to unions of conjunctive queries (instead of first-order queries), but in exchange consider additional knowledge encoded through an ontology.

Table 7.4: The probabilistic database $\mathcal{P}_v$ (repeated from Table 5.2).

Vegetarian		FriendOf			Eats			Meat	
alice	0.7	alice	bob	0.7	bob	spinach	0.7	shrimp	0.7
bob	0.9	alice	chris	0.8	chris	mussels	0.8	mussels	0.9
chris	0.6	bob	chris	0.1	alice	broccoli	0.2	seahorse	0.3

Importantly, in MPD (respectively, MPH), we are interested in finding the database (respectively the hypothesis) that maximizes the query probability. In the presence of ontological rules, we need to ensure that the chosen database is at the same time consistent with the ontology. More precisely, for ontology-mediated queries, models must be consistent with the ontology; thus, the definitions of MPD and MPH are adapted accordingly.

In our analysis, we also include the case where only negative constraints are allowed, as this is closest to the constraints over PDBs that we described in Chapter 5 and it gives us a baseline for our study. We then provide a thorough complexity analysis based on Datalog$^{\pm}$ languages for both problems.

7.3.1 The Most Probable Database Problem for Ontological Queries

In the most probable database problem for ontological queries, we consider only the *consistent* worlds induced by the PDB, and thus maximize only over consistent worlds.

Definition 7.20 Let $\mathcal{P}$ be a probabilistic database and Q a query. The *most probable database* for an OMQ (Q, Σ) over a PDB $\mathcal{P}$ is given by

$$\underset{\mathcal{D} \models (Q,\Sigma), mods(\mathcal{D},\Sigma) \neq \emptyset}{\arg\max} \mathrm{P}(\mathcal{D}),$$

where $\mathcal{D}$ ranges over all worlds induced by $\mathcal{P}$. ◊

To illustrate the semantics, we now revisit the Example 5.8 and the PDB $\mathcal{P}_v$, which is depicted in Table 7.4.

Example 7.21 Recall the following query from Chapter 5

$$Q_{\mathsf{veg}} := \forall x, y \ \neg \mathsf{Vegetarian}(x) \vee \neg \mathsf{Eats}(x, y) \vee \neg \mathsf{Meat}(y).$$

The most probable database for Q_{veg} contains all atoms from $\mathcal{P}_v$ that have a probability above 0.5, except for Vegetarian(chris). We can impose the same constraint through the negative constraint

$$\forall x, y \ \mathsf{Vegetarian}(x) \wedge \mathsf{Eats}(x, y) \wedge \mathsf{Meat}(y) \rightarrow \bot,$$

and then the most probable database for the query $\top$ will be the same as before. Obviously, we can additionally impose constraints in the form of TGDs such as

$$\forall x \ \mathsf{Vegetarian}(x) \rightarrow \exists y \ \mathsf{FriendOf}(x, y) \wedge \mathsf{Vegetarian}(y),$$

which states that vegetarians have friends who are themselves vegetarians. Note, however, that OMQs operate over infinite domains, whereas first-order queries in databases are defined over finite domains and thus these queries are not comparable in terms of expressivity. ◊

The corresponding decision problem is defined as before with the only difference that now we consider ontology-mediated queries.

Definition 7.22 (MPD) Let (Q, Σ) be an OMQ, $\mathcal{P}$ a probabilistic database and $p \in (0, 1]$ a threshold. MPD is the problem of deciding whether there exists a database $\mathcal{D}$ that entails (Q, Σ) with $\mathrm{P}(\mathcal{D}) > p$. MPD is parametrized with the language of the ontology and the query; we write $\mathsf{MPD}(\mathsf{Q}, \mathcal{L})$ to define MPD on the class Q of queries and on the class of ontologies restricted to the languages $\mathcal{L}$ ◊

We start our complexity analysis with some simple observations. Note first that consistency of a database $\mathcal{D}$ with respect to a program Σ can be written as $\mathcal{D} \not\models (\bot, \Sigma)$, or equivalently, $\mathcal{D} \not\models (Q_\bot, \Sigma^+)$, where $Q_\bot$ is the UCQ obtained from the disjunction of the bodies of all NCs in Σ, and Σ^+ is the corresponding positive program (that contains all TGDs of Σ). This transformation allows us to rewrite the NCs into the query.

A naïve approach to solve MPD is to first guess a database $\mathcal{D}$, and then check that it entails the given OMQ, does *not* entail the query $(Q_\bot, \Sigma^+)$ (i.e., it is consistent with the program), and exceeds the probability threshold. Since the probability can be computed in polynomial time, the problem can be decided by a nondeterministic Turing machine using an oracle to check OMQA. Obviously MPD is at least as hard as OMQA in the underlying ontology languages. These observations result in the following Theorem.

Theorem 7.23 *Let $\mathfrak{C}$ denote the data (respectively, fixed-program, bounded-arity, combined) complexity of ontology-mediated query answering for a Datalog$^{\pm}$ language $\mathcal{L}$. $\mathsf{MPD}(\mathsf{UCQ}, \mathcal{L})$ is $\mathfrak{C}$-hard and in $\mathrm{NP}^{\mathfrak{C}}$ under the same complexity assumptions.*

By a reduction from 3-colorability, we show that MPD is NP-hard already in data complexity for OMQs, even if we only use NCs, i.e., the query and the positive program Σ^+ are empty. This strengthens our previous result about ∀FO queries (from Chapter 5), since NCs can be expressed by universal queries, but are not allowed to use negated atoms.

Theorem 7.24 $\mathsf{MPD}(\mathsf{UCQ}, \mathsf{NC})$ *is* NP*-hard in data complexity (which holds even for instance queries).*

Proof. We provide a reduction from the well-known 3-colorability problem: given an undirected graph $G = (V, E)$, decide whether the nodes of G are 3-colorable. We first define the PDB $\mathcal{P}_G$ as follows. For all edges $(u, v) \in E$, we add the atom $\mathsf{E}(u, v)$ with probability 1, and for all nodes $u \in V$, we add the atoms $\mathsf{V}(u, 1)$, $\mathsf{V}(u, 2)$, $\mathsf{V}(u, 3)$, each with probability 0.7. In this encoding, the atoms $\mathsf{V}(u, 1)$, $\mathsf{V}(u, 2)$, $\mathsf{V}(u, 3)$ correspond to different colorings of the same node u.

We next define the conditions for 3-colorability through a set Σ containing only negative constraints (that do not depend on G). We need to ensure that each node is assigned *at most* one color, which is achieved by means of the negative constraints:

$$\mathsf{V}(x, 1), \mathsf{V}(x, 2) \to \bot, \quad \mathsf{V}(x, 1), \mathsf{V}(x, 3) \to \bot, \quad \mathsf{V}(x, 2), \mathsf{V}(x, 3) \to \bot$$

Similarly, we need to enforce that the neighboring nodes are not assigned the same color, which we ensure with the negative constraint:

$$\mathsf{E}(x, y), \mathsf{V}(x, c), \mathsf{V}(y, c) \to \bot$$

Finally, we define the query $Q \coloneqq \top$ and prove the following claim.

Claim. G is 3-colorable if and only if there exists a database $\mathcal{D}$ such that $\mathcal{D} \not\models (\bot, \Sigma)$ and $\mathrm{P}(\mathcal{D}) \geq (0.7 \cdot 0.3 \cdot 0.3)^{|V|}$.

Suppose that there exists a database $\mathcal{D}$ with a probability of at least $(0.7 \cdot 0.3 \cdot 0.3)^{|V|}$ that satisfies all NCs in Σ. Then, for every node $u \in V$, $\mathcal{D}$ must contain exactly one tuple $\mathtt{V}(u, c)$ for some color $c \in \{1, 2, 3\}$. Recall that *at most* was ensured by the first three NCs; hence, in order to achieve the given threshold, *at least* one of these tuples must be present. This yields a unique coloring for the nodes. Furthermore, since $\mathcal{D}$ must contain all tuples corresponding to the edges of G, and $\mathcal{D}$ satisfies the last NC, we conclude that G is 3-colorable.

Suppose that G is 3-colorable. Then, for a valid coloring, we define a DB $\mathcal{D}$ that contains all tuples that correspond to the edges, and add all tuples $\mathtt{V}(u, c)$ where c is the color of u. It is easy to see that $\mathcal{D}$ is consistent with Σ and $\mathrm{P}(\mathcal{D}) = (0.7 \cdot 0.3 \cdot 0.3)^{|V|}$, which concludes the proof. □

We show a matching upper bound (which holds even in fixed-program complexity), by reconsidering the approach from Theorem 7.23.

Theorem 7.25 *Let $\mathcal{L}$ be a Datalog$^{\pm}$ language, where ontology-mediated query answering is in* NP *in fixed-program complexity and in* P *in the data complexity. Then,* $\mathsf{MPD}(\mathsf{UCQ}, \mathcal{L})$ *is in* NP *in fixed-program complexity.*

Proof. Note that the query $(Q_\bot, \Sigma^+)$ is fixed; thus, for the non-entailment check to verify consistency, we can refer to the *data complexity* of OMQ entailment. Since this is possible in P, and further the fixed-program complexity does not exceed NP, the whole test can be done by a single nondeterministic Turing machine in polynomial time. □

This result yields an upper bound of NP for most of the classes we consider for MPD. Under bounded-arity complexity assumptions, we observe an increase in complexity, which intuitively comes from the fact that the query $(Q_\bot, \Sigma^+)$ is not fixed anymore.

Theorem 7.26 $\mathsf{MPD}(\mathsf{UCQ}, \mathsf{NC})$ *is Σ_2^{P}-hard in bounded-arity complexity (which holds even for atomic queries).*

Proof. Consider the formula

$$\Phi = \exists x_1, \ldots, x_n \, \forall y_1, \ldots, y_m \, \varphi,$$

where $\varphi = \varphi_1 \vee \cdots \vee \varphi_k$ is a propositional formula in 3DNF. We encode the truth values of the variables x_i using the atoms $\langle \mathsf{V}(x_i, 0) : 0.9 \rangle$, $\langle \mathsf{V}(x_i, 1) : 0.9 \rangle$ in the PDB $\mathcal{P}_\Phi$, which are constrained by the NC $\mathsf{V}(x, 0) \wedge \mathsf{V}(x, 1) \to \bot$. The idea is that the worlds $\mathcal{D}$ with maximal probability must satisfy exactly one of $\mathsf{V}(x_i, 0)$, $\mathsf{V}(x_i, 1)$ for each x_i, thereby representing a truth value assignment for $\vec{x}$.

We explain the encoding of the conjunctions in Φ on the example of $\varphi_j = x_1 \wedge \neg y_4 \wedge \neg x_3$. We introduce a predicate $\mathsf{C}_j(t, y_4)$ that describes the truth value t of φ_j as a function of the truth value of y_4, which of course depends on the truth values chosen for x_1 and x_3 via the V-atoms. For example, if y_4 is true, then φ_j must be false, which is expressed by the tuple $\mathsf{C}_j(0, 1)$. As above, we add all tuples $\langle \mathsf{C}_j(t, y_4) : 0.9 \rangle$ with $t, y_4 \in \{0, 1\}$ to $\mathcal{P}_\Phi$. The idea is again that, for a certain value of y_4, exactly one of the two atoms $\mathsf{C}_j(0, y_4)$, $\mathsf{C}_j(1, y_4)$ will be true in the chosen world $\mathcal{D}$ (not both and not neither).

This behavior is enforced by a series of NCs. For example, if x_1 is true and x_3 is false, then $\mathsf{C}_j(1, 0)$ must be true, i.e., $\mathsf{C}_j(0, 0)$ must be false:

$$\mathsf{V}(x_1, 1) \wedge \mathsf{V}(x_3, 0) \wedge \mathsf{C}_j(0, 0) \to \bot$$

Moreover, making y_4 true will always make the conjunction false, regardless of the values of x_1 and x_3:

$$\mathsf{C}_j(1, 1) \to \bot$$

Finally, if either x_1 is false or x_3 is true, then it is impossible to satisfy φ_j, regardless of the value of y_4:

$$\mathsf{V}(x_1, 0) \wedge \mathsf{C}_j(1, y_4) \to \bot$$
$$\mathsf{V}(x_3, 1) \wedge \mathsf{C}_j(1, y_4) \to \bot$$

These four NCs together ensure that, in every world $\mathcal{D}$ of maximal probability (which determines a fixed truth value assignment for $x_1, \ldots, x_n$), the satisfied tuples $\mathsf{C}_j(t, y_4)$ determine the truth value of φ_j as a function of the truth value of y_4. For example, if $\mathcal{D}$ satisfies $\mathsf{V}(x_1, 0)$ and $\mathsf{V}(x_3, 1)$ (and hence neither $\mathsf{V}(x_1, 1)$ nor $\mathsf{V}(x_3, 0)$), then it must satisfy also $\mathsf{C}_j(0, 0)$ and $\mathsf{C}_j(0, 1)$ (but neither $\mathsf{C}_j(1, 0)$ nor $\mathsf{C}_j(1, 1)$).

In general, the predicate C_j is of arity $1+|\mathbf{y}_j|$, where $\mathbf{y}_j$ are the variables from $y_1, \ldots, y_m$ that occur in φ_j. As φ_j contains 3 literals, this implies means that the arity can be at most 4. This results in atoms of the form $\mathsf{C}_j(t, \mathbf{y}_j)$. For each clause, four NCs like the ones above enforce that exactly one of $\mathsf{C}_j(0, \mathbf{y}_j)$ and $\mathsf{C}_j(1, \mathbf{y}_j)$ is true, for each valuation of $\mathbf{y}_j$, and this describes exactly the behavior of φ_j. Finally, we add the NC

$$\bigwedge_{j=1}^{k} \mathsf{C}_j(0, \mathbf{y}_j) \to \bot$$

to express that every valuation of the variables $y_1, \ldots, y_m$ must satisfy at least one conjunction φ_j. Hence, the existence of a consistent world $\mathcal{D}$ with $\mathrm{P}(\mathcal{D}) \geq (0.09)^\ell$, where $\ell = n + \sum_{j=1}^{k} 2^{|\mathbf{y}_j|}$, is equivalent to the validity of Φ. Since ℓ is bounded by $n + 8k$, the threshold can be written using linearly many bits. Observe that we do not need a query (we can choose $Q = \top$) nor any TGDs, and the maximal arity of the used predicates is 4. □

We obtain a matching upper bound from Theorem 7.23 if $\mathfrak{C} = \mathrm{NP}$. Most of the remaining hardness results follow from the complexity of classical OMQA, since an OMQ (Q, Σ) is entailed by a consistent database $\mathcal{D}$ if and only if the most probable database with respect to $\{\langle t : 1 \rangle \mid t \in \mathcal{D}\}$ has probability 1. In combination with Theorem 7.23, this yields tight complexity results for large deterministic classes like PSpace or Exp.

This leaves open only the case of the class A, for which OMQA is NExp-complete in bounded-arity complexity, which yields an upper bound of $\mathrm{NP}^{\mathrm{NExp}} = \mathrm{P}^{\mathrm{NExp}} = \mathrm{P}^{\mathrm{NE}}$ due to Theorem 7.23 and results from (Hemachandra 1989). We show that this bound is tight, using a reduction from a $\mathrm{P}^{\mathrm{NExp}}$-complete version of the *tiling problem* (Fürer 1983).

Theorem 7.27 $\mathsf{MPD}(\mathsf{UCQ}, \mathsf{A})$ *is* P^{NE}*-hard in ba-combined complexity.*

Proof. We give a reduction from an extended tiling problem. Recall that, the original *tiling problem* (Fürer 1983) is defined as follows: Let T be a set of square tile types, $H, V \subseteq T \times T$ be the horizontal and vertical compatibility relations, respectively, and n be an integer in unary. A $2^n \times 2^n$ tiling is a function

$$f \colon \{1, \dots, 2^n\} \times \{1, \dots, 2^n\} \to T$$

such that $(f(i,j), f(i,j+1)) \in H$ and $(f(i,j), f(i+1,j)) \in V$, for each i and j. An instance of the tiling problem is a tuple (T, H, V, n), and the decision problem is to determine whether a $2^n \times 2^n$ tiling exists.

We give a reduction from the following *extended tiling problem* (Eiter, Lukasiewicz, and Predoiu 2016), which is $\mathrm{P}^{\mathrm{NExp}}$-complete: Given a triple $(m, \mathsf{TP}_1, \mathsf{TP}_2)$ of an integer m in unary and tiling problems TP_1 and TP_2 for the exponential square $2^n \times 2^n$, does, for every initial condition $w = w_1 \dots w_m$ for the first row, TP_1 have no solution with w, or does TP_2 have some solution with w?

The construction makes use of the result that any instance TP of tiling the $2^n \times 2^n$-square, given n, relations H and V, and an initial tiling condition $w = w_1 \dots w_m$, is reducible to OMQ answering with acyclic TGDs in polynomial time such that the query $(\mathsf{Tiling}, \Sigma^{\mathsf{TP},|w|})$ is entailed by $\mathcal{D}^{\mathsf{TP}} \cup \mathcal{D}^w$, where $\Sigma^{\mathsf{TP},|w|}$ is constructed from TP and $|w|$, $\mathcal{D}^{\mathsf{TP}}$ from TP, and $\mathcal{D}^w = \{\mathsf{Init}_j(w_j) \mid 1 \leq j \leq m\}$, if and only if TP has a solution with w.

We define the PDB $\mathcal{P}$ as the set of all atoms $\langle t : 1 \rangle$ such that $f \in \mathcal{D}^{\mathsf{TP}_1} \cup \mathcal{D}^{\mathsf{TP}_2}$, all atoms $\langle \mathsf{Init}_j(d) : 0.5 \rangle$ such that $d \in T$ and $1 \leq j \leq m$, and the atoms $\langle \mathsf{Tiling}_2(0) : 0.5 \rangle$ and $\langle \mathsf{Tiling}_2(1) : 0.5 \rangle$.

We define the program Σ as the union of $\Sigma_r^{\mathsf{TP}_1,|w|}$ and $\Sigma_r^{\mathsf{TP}_2,|w|}$, obtained from $\Sigma^{\mathsf{TP}_1,|w|}$ and $\Sigma^{\mathsf{TP}_2,|w|}$ by renaming all derived predicates P to P_1 and P_2, respectively, and by replacing Tiling_2 by $\mathsf{Tiling}_2(1)$, together with the NCs $\mathsf{Tiling}_2(0), \mathsf{Tiling}_2(1) \to \bot$ and all $\mathsf{Init}_j(d), \mathsf{Init}_j(d') \to \bot$ such that $d, d' \in T$ and $1 \leq j \leq m$.

Then, with the UCQ Q defined as

$$\exists x_1, \dots, x_m \bigwedge_{j=1}^{m} \mathsf{Init}_j(x_j) \wedge \mathsf{Tiling}_1 \wedge \mathsf{Tiling}_2(0),$$

we obtain that there is a world induced by $\mathcal{P}$ that satisfies (Q, Σ) if and only if $(m, \mathsf{TP}_1, \mathsf{TP}_2)$ is a no-instance of the extended tiling problem. □

We note that the modified tiling problem is novel and has been recently introduced by (Eiter, Lukasiewicz, and Predoiu 2016) as a canonical $\mathrm{P}^{\mathrm{NExp}}$-complete problem. This concludes our complexity analysis of MPD for OMQs.

Table 7.5: Complexity results for MPD for ontology-mediated queries: Note that the result for ∀FO queries in PDBs has been strengthened towards the weaker representation NC.

Datalog$^{\pm}$ Languages	Most Probable Database			
	data	fixed-program	bounded-arity	combined
NC	NP	NP	Σ_2^{P}	Σ_2^{P}
L, LF, AF	NP	NP	Σ_2^{P}	PSPACE
S, SF	NP	NP	Σ_2^{P}	EXP
A	NP	NP	P^{NE}	P^{NE}
GF, F	NP	NP	Σ_2^{P}	EXP
G	NP	NP	EXP	2EXP
WS, WA	NP	NP	2EXP	2EXP
WG	EXP	EXP	EXP	2EXP

Overview of the Results

The complexity results for MPD are summarized in Table 7.5. Importantly, we have included the class NC in the analysis. In Theorem 7.24, we have shown NP-hardness in data complexity already for MPD(UCQ, NC) by a reduction from 3-colorability. Note that this result strengthens the data complexity result obtained for MPD for ∀FO queries. Moreover, since any general Datalog$^{\pm}$ program contains negative constraints, Theorem 7.24 already implies lower complexity bounds for all classes in data complexity. In all of the cases this turns out to be also a matching lower bound except for the case of class WG. The precise complexity of MPD(UCQ, WG) follows from Theorem 7.23.

Interestingly, all NP upper bounds from the data complexity apply to the fixed-program complexity. This is mainly due to the following observation: the consistency check can still be done in polynomial time in these classes in fixed-program complexity. Therefore, for all languages where OMQA is NP-complete, we observe that the complexity remains the same for MPD in fixed-program complexity.

This picture changes significantly for the bounded-arity complexity. For classes where OMQA is NP-complete in the bounded-arity, we additionally need to make a coNP check to decide consistency of the world that is chosen. We therefore observe a Σ_2^{P} upper bound for the most probable database problem with respect to these classes. Hardness holds already for programs that contain only negative constraints as shown in Theorem 7.26. All other results regarding the deterministic complexity classes are a consequence of Theorem 7.23. The class A is again an interesting case. OMQA is NEXP-complete and consistency checking is coNEXP-complete for A. Thus, the problem can be solved in $\mathrm{NP}^{\mathrm{NEXP}}$, which is equivalent to P^{NE} by (Hemachandra 1989). We prove a matching lower bound by a reduction from an extended tiling problem.

7.3.2 The Most Probable Hypothesis Problem for Ontological Queries

In the most probable hypothesis problem for ontological queries, we have to update the definition of most probable hypothesis to take into account only consistent worlds.

Definition 7.28 The *most probable hypothesis* for an OMQ (Q, Σ) over a PDB $\mathcal{P}$ is

$$\arg\max_{\mathcal{H} \models (Q,\Sigma)} \sum_{\substack{\mathcal{D} \supseteq \mathcal{H} \\ mods(\mathcal{D},\Sigma) \neq \emptyset}} \mathrm{P}(\mathcal{D}),$$

where $\mathcal{H}$ is a set of (non-probabilistic) atoms t occurring in $\mathcal{P}$. ◊

The corresponding decision problem is defined as before, but parametrized with ontology-mediated queries.

Definition 7.29 (MPH) Let (Q, Σ) be an OMQ, $\mathcal{P}$ a PDB, and $p \in (0, 1]$ a threshold. MPH is the problem of deciding whether there exists a hypothesis $\mathcal{H}$ that satisfies (Q, Σ) with $\mathrm{P}(\mathcal{H}) > p$. MPH is parametrized with the language of the ontology and the query; we write $\mathsf{MPH}(\mathsf{Q}, \mathcal{L})$ to define MPH on the class Q of queries and on the class of ontologies restricted to the languages $\mathcal{L}$. ◊

Importantly, computing the probability of a hypothesis for an OMQ is not as easy as for classical database queries since now there are also inconsistent worlds that shall be ruled out. As a consequence, computing the probability of a hypothesis becomes PP-hard, which makes a significant difference in the complexity analysis.

To solve MPH for OMQs, one can guess a hypothesis, and then check whether it entails the query, and whether the probability mass of its consistent extensions exceeds the given threshold. The latter part can be done by a PP Turing machine with an oracle for OMQA. The oracle can be used also for the initial entailment check. Clearly, $\mathsf{MPH}(\mathsf{UCQ}, \mathcal{L})$ is at least as hard as OMQA in $\mathcal{L}$. The next theorem follows from these observations.

Theorem 7.30 *Let $\mathfrak{C}$ denote the data (respectively, fixed-program, bounded-arity, combined) complexity of ontology-mediated query answering for a Datalog$^{\pm}$ language $\mathcal{L}$.* $\mathsf{MPD}(\mathsf{UCQ}, \mathcal{L})$ *is $\mathfrak{C}$-hard and in* $\mathrm{NP}^{\mathrm{PP}^{\mathfrak{C}}}$ *under the same complexity assumptions.*

For any $\mathfrak{C} \subseteq \mathrm{PH}$, this result yields $\mathrm{NP}^{\mathrm{PP}^{\mathrm{PH}}} = \mathrm{NP}^{\mathrm{P}^{\mathrm{PP}}} = \mathrm{NP}^{\mathrm{PP}}$ as an upper bound, due to a result from Toda 1989. Except for PP, all other upper bounds in Table 7.6 also follow from this observation. Note that for $\mathfrak{C} = \mathrm{NEXP}$, the whole $\mathrm{PP}^{\mathrm{NEXP}}$ computation can be done by an NEXP oracle, and hence we again obtain $\mathrm{NP}^{\mathrm{NEXP}} = \mathrm{P}^{\mathrm{NE}}$ in this case. On the other hand, we can transfer most of the lower bounds from MPD by a simple reduction.

Theorem 7.31 $\mathsf{MPH}(\mathsf{UCQ}, \mathcal{L})$ *is at least as hard as* $\mathsf{MPD}(\mathsf{UCQ}, \mathcal{L})$ *in fixed-program, bounded-arity and combined complexity.*

Proof. Consider an OMQ (Q, Σ) over a PDB $\mathcal{P}$, for which we want to find a consistent world of probability greater than q. We enforce that the most probable hypothesis needs to make a choice for all atoms in $\mathcal{P}$, by replacing each atom $\langle \mathsf{L}(\vec{u}) : p \rangle$ by two new atoms $\langle \mathsf{F}(\vec{u}, 1) : p \rangle$ and $\langle \mathsf{F}(\vec{u}, 0) : 1 - p \rangle$, where F is a fresh predicate that increases the arity of L by one, and 0 and 1 are fresh constants. We denote the resulting PDB by $\mathcal{P}'$. As the query, we use

$$Q' := Q^1 \wedge \bigwedge_{\langle \mathsf{L}(\vec{u}):p \rangle \in \mathcal{P}} \exists z \, \mathsf{F}(\vec{u}, z),$$

where Q^1 is obtained from Q by replacing all atoms $\mathsf{L}(\vec{x})$ by $\mathsf{F}(\vec{x}, 1)$. The query Q' is equivalent to a UCQ that is of size polynomial in the size of Q and $\mathcal{P}$.

Finally, in the program Σ we similarly replace each tuple $\mathsf{L}(\vec{x})$ by $\mathsf{F}(\vec{x}, 1)$, and add the NCs $\mathsf{F}(\vec{x}, 1) \wedge \mathsf{F}(\vec{x}, 0) \rightarrow \bot$ for each of the new predicates F. The resulting program Σ' still satisfies the constraints of fixed-program/bounded-arity combined complexity, and is in the same class as Σ. We prove the following claim.

Claim. Most probable hypothesis for (Q', Σ') over $\mathcal{P}'$ exceeds the threshold q^2 if and only if the most probable database for (Q, Σ) over $\mathcal{P}$ exceeds q.

Assume that the latter is the case, and the database $\mathcal{D}$ has a probability of $d > q$ w.r.t. $\mathcal{P}$. Then the hypothesis that contains all atoms $\mathsf{F}(\vec{u}, 1)$ for which $\mathsf{L}(\vec{u}) \in \mathcal{D}$, and $\mathsf{F}(\vec{u}, 0)$ whenever $\mathsf{L}(\vec{u}) \notin \mathcal{D}$ has the probability $d^2 > q^2$ over $\mathcal{P}'$ and satisfies the OMQ (Q', Σ').

Conversely, assume that the hypothesis $\mathcal{H}$ for (Q', Σ') over $\mathcal{P}'$ has a probability $d > q^2$. Since $\mathcal{H}$ satisfies Q' and the new NCs in Σ', for each tuple $\langle \mathsf{L}(\vec{u}) : p \rangle \in \mathcal{P}$, it must contain exactly one of the atoms $\mathsf{F}(\vec{u}, 0)$ or $\mathsf{F}(\vec{u}, 1)$, which together contribute a probability of $(1-p)^2$ or p^2, respectively, over $\mathcal{P}'$. Hence, the sum in Definition 7.28 collapses to one element, which corresponds exactly to the database $\mathcal{D}$ obtained by collecting all atoms $\mathsf{L}(\vec{u})$ for which $\mathsf{F}(\vec{u}, 1)$ is in $\mathcal{H}$. This database has a probability of $\sqrt{d} > q$ over $\mathcal{P}$, and must satisfy the original query and NCs. □

In data complexity, our results for MPH turn out to be relatively surprising. For FO-rewritable Datalog$^{\pm}$ languages, we immediately obtain an $\mathrm{NP}^{\mathrm{PP}}$ upper bound for MPH as a consequence of Theorem 7.30. The obvious question is whether this is also a matching lower bound for FO-rewritable languages?

In general, the (oracle) test of checking whether the probability of an hypothesis exceeds the threshold value is PP-hard. Thus, PP-hardness for MPH for FO-rewritable languages can not be avoided in general. The remaining question is whether the hypothesis can be identified efficiently, or do we really need to guess the hypothesis? Fortunately, this can be avoided since we can walk through polynomially many hypothesis (in data complexity) and combine the threshold tests for each of the hypothesis into single PP computation using the results of (Beigel, Reingold, and Spielman 1995).

Theorem 7.32 *Let $\mathcal{L}$ be a Datalog$^{\pm}$ languages, for which ontology-mediated query answering is in AC^0 in data complexity. Then, $\mathsf{MPH}(\mathsf{UCQ}, \mathcal{L})$ is PP-complete in data complexity under polynomial time Turing reductions.*

Proof. PP-hardness follows from the complexity of probabilistic query evaluation over PDBs (Suciu et al. 2011, Corollary 3.18) since we can choose $Q = \top$ and reformulate any UCQ into a set of NCs such that the consistency of a database is equivalent to the non-satisfaction of the UCQ.

We consider an OMQ (Q, Σ), a PDB $\mathcal{P}$, and a threshold p. Since the query is FO-rewritable, it is equivalent to an ordinary UCQ Q_Σ over $\mathcal{P}$. Similarly, we can rewrite the UCQ $Q_\bot$ expressing the non-satisfaction of the NCs into a UCQ $Q_{\bot,\Sigma}$. By the observation in Theorem 5.22 we can enumerate all hypotheses $\mathcal{H}$, which are the polynomially many matches for Q_Σ in $\mathcal{P}$, and then have to check for each $\mathcal{H}$ whether the probability of all consistent extensions exceeds p. The latter part is equivalent to evaluating $\neg Q_{\bot,\Sigma} \wedge \bigwedge_{t \in \mathcal{H}} t$

over $\mathcal{P}$, which can be done by a PP oracle. We accept if and only if one of these PP checks yields a positive answer. In the terminology of (Beigel, Reingold, and Spielman 1995), this is a polynomial-time disjunctive reduction of our problem to a PP problem. Since that paper shows that PP is closed under such reductions, we obtain the desired PP upper bound. □

Given Theorem 7.32, one may wonder whether the PP upper bound for MPH also applies to languages where OMQA is P-complete in data complexity. In a surprising result, we show that MPH(UCQ, GF) is already $\mathrm{NP}^{\mathrm{PP}}$-hard.

Theorem 7.33 MPH(UCQ, GF) *is* $\mathrm{NP}^{\mathrm{PP}}$*-hard in data complexity.*

Proof. We reduce the following problem from (Wagner 1986): decide the validity of

$$\Phi = \exists x_1, \ldots, x_n \; \mathsf{C}^c \, y_1, \ldots, y_m \, \varphi,$$

where $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_k$ is a propositional formula in CNF, defined over the variables $x_1, \ldots, x_n, y_1, \ldots, y_m$. This amounts to checking whether there is a partial assignment for $x_1, \ldots, x_n$ that admits at least c extensions to $y_1, \ldots, y_m$ that satisfy φ.

We can assume without loss of generality that each of the clauses φ_i contains exactly three literals: shorter clauses can be padded by copying existing literals, and longer clauses can be abbreviated using auxiliary variables that are included under the counting quantifier C^c. Since the values of these variables are uniquely determined by the original variables, this does not change the number of satisfying assignments.

We define the PDB $\mathcal{P}_\Phi$ that describes the structure of Φ:

- For each variable v occurring in Φ, $\mathcal{P}_\Phi$ contains the tuples $\langle \mathsf{V}(v, 0) : 0.5 \rangle$ and $\langle \mathsf{V}(v, 1) : 0.5 \rangle$, where v is viewed as a constant. These tuples represent the assignments that map v to *false* and *true*, respectively.
- For each clause φ_j, we introduce the tuple $\langle \mathsf{C}(v_1, t_1, v_2, t_2, v_3, t_3) : 1 \rangle$, where t_i is 1 if v_i occurs negatively in φ_j, and 0 otherwise (again, all terms are constants). For example, for $x_3 \vee \neg y_2 \vee x_7$, we use the tuple $\langle \mathsf{C}(x_3, 0, y_2, 1, x_7, 0) : 1 \rangle$. This encodes the knowledge about the partial assignments that do not satisfy φ.
- We use auxiliary atoms $\langle \mathsf{A}(x_1) : 1 \rangle$, $\langle \mathsf{S}(x_1, x_2) : 1 \rangle$, …, $\langle \mathsf{S}(x_{n-1}, x_n) : 1 \rangle$, $\langle \mathsf{L}(x_n) : 1 \rangle$ to encode the order on the variables x_i, and similarly for y_j we use the atoms $\langle \mathsf{B}(y_1) : 1 \rangle$, $\langle \mathsf{S}(y_1, y_2) : 1 \rangle$, …, $\langle \mathsf{S}(y_{m-1}, y_m) : 1 \rangle$, and $\langle \mathsf{L}(y_m) : 1 \rangle$.

We now describe the program Σ used for the reduction. First, we detect whether all variables x_i $(1 \leq i \leq n)$ have a truth assignment (i.e., at least one of the facts $\mathsf{V}(x_i, 0)$ or $\mathsf{V}(x_i, 1)$ is present) by the special nullary predicate A, using the auxiliary unary predicates V and A:

$$\begin{aligned} \mathsf{V}(x, t) &\to \mathsf{V}(x) \\ \mathsf{A}(x) \wedge \mathsf{V}(x) \wedge \mathsf{S}(x, x') &\to \mathsf{A}(x'), \\ \mathsf{A}(x) \wedge \mathsf{V}(x) \wedge \mathsf{L}(x) &\to \mathsf{A}, \end{aligned}$$

where x, x', t are variables. We do the same for the variables $y_1, \ldots, y_m$:

$$\begin{aligned} \mathsf{B}(y) \wedge \mathsf{V}(y) \wedge \mathsf{S}(y, y') &\to \mathsf{B}(y'), \\ \mathsf{B}(y) \wedge \mathsf{V}(y) \wedge \mathsf{L}(y) &\to \mathsf{B}. \end{aligned}$$

Now, the query $Q = \mathsf{A}$ ensures that only such hypotheses are valid that at least contain a truth assignment for the variables $x_1, \ldots, x_n$.

Next, we restrict the assignments to satisfy φ by using additional NCs in Σ. First, we ensure that there is no "inconsistent" assignment for any variable v, i.e., only one of the facts $\mathsf{V}(v, 0)$ or $\mathsf{V}(v, 1)$ holds:

$$\mathsf{V}(v, 0) \wedge \mathsf{V}(v, 1) \rightarrow \bot$$

Furthermore, if all variables $y_1, \ldots, y_m$ have an assignment, then none of the clauses in φ can be falsified:

$$\mathsf{C}(v_1, t_1, v_2, t_2, v_3, t_3) \wedge \mathsf{V}(v_1, t_1) \wedge \mathsf{V}(v_2, t_2) \wedge \mathsf{V}(v_3, t_3) \wedge \mathsf{B} \rightarrow \bot,$$

where $v_1, t_1, v_2, t_2, v_3, t_3$ are variables. We now prove the following claim.

Claim. Φ is valid if and only if there exists a hypothesis $\mathcal{H}$ that satisfies (Q, Σ) such that all consistent databases that extend $\mathcal{H}$ sum up to a probability (under $\mathcal{P}_\Phi$) of at least $p = 0.25^n \cdot 0.25^m(3^m - 2^m + c)$.

Assume that such a hypothesis $\mathcal{H}$ exists. Since $\mathcal{H} \models (Q, \Sigma)$, we know that for each x_i $(1 \leq i \leq n)$ one of the atoms $\mathsf{V}(x_i, 0)$, $\mathsf{V}(x_i, 1)$ is included in $\mathcal{H}$. In each consistent extension of $\mathcal{H}$, it must be the case that the complementary facts (representing an inconsistent assignment for x_i) are false. In particular, these complementary facts cannot be part of $\mathcal{H}$ since then its probability would be 0. Hence, we can ignore the factor 0.25^n in the following. There are exactly $3^m - 2^m$ databases satisfying $\mathcal{H}$ that represent consistent, but incomplete assignments for the variables y_j. Since these databases do not entail B, they are all consistent, and hence counted towards the total sum. The inconsistent assignments for $y_1, \ldots, y_m$ yield inconsistent databases, which leaves us only with the 2^m databases representing proper truth assignments. Those that violate at least one clause of φ become inconsistent, and hence there are at least c such consistent databases if and only if there are at least c extensions of the assignment represented by $\mathcal{H}$ that satisfy φ. We conclude these arguments by noting that the probability of each individual choice of atoms $\mathsf{V}(y_j, t_j)$ $(1 \leq j \leq m)$ is 0.25^m.

On the other hand, if Φ is valid, then we can use the same arguments to construct a hypothesis $\mathcal{H}$ (representing the assignment for $x_1, \ldots, x_n$) that exceeds the given threshold. □

We observe a sharp contrast in data complexity results for MPH: PP vs $\mathrm{NP}^{\mathrm{PP}}$, as summarized in Theorems 7.32 and 7.33, respectively. This difference disappears once we consider fixed-program complexity.

Theorem 7.34 $\mathsf{MPH}(\mathsf{UCQ}, \mathsf{NC})$ *is* $\mathrm{NP}^{\mathrm{PP}}$*-hard in fixed-program complexity and in bounded-arity complexity.*

Proof. We again consider the formula

$$\Phi = \exists x_1, \ldots, x_n \; \mathsf{C}^c \, y_1, \ldots, y_m \, \varphi,$$

where $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_k$ is in 3CNF and the PDB $\mathcal{P}_\Phi$ is defined as follows:

– For each variable x_i, $1 \leq i \leq n$, we use the atoms $\langle \mathsf{V}(x_i, 0) : 0.5 \rangle$ and $\langle \mathsf{V}(x_i, 1) : 0.5 \rangle$, and for y_j, $1 \leq j \leq m$, we use $\langle \mathsf{V}(y_j, 0) : p \rangle$ and $\langle \mathsf{V}(y_j, 1) : p \rangle$, where p is a fixed, large probability that we specify later.

– For each clause φ_j, we introduce the atom $\langle \mathsf{C}(v_1, t_1, v_2, t_2, v_3, t_3) : 1 \rangle$ as before.

We then define the query

$$Q_\Phi = \exists t_1, \ldots, t_n \, \mathsf{V}(x_1, t_1) \wedge \cdots \wedge \mathsf{V}(x_n, t_n),$$

in order to enforce that any hypothesis contains a truth assignment for the existentially quantified variables. For the variables $y_1, \ldots, y_m$, this is handled by a special choice of p, which ensures that the probabilities of the incomplete assignments sum up to a value that is smaller than the probability of a single complete assignment; hence, we can ignore the incomplete assignments when counting the complete assignments. The program Σ hence only needs to contain the two NCs

$$\mathsf{V}(v, 0) \wedge \mathsf{V}(v, 1) \rightarrow \bot$$
$$\mathsf{C}(v_1, t_1, v_2, t_2, v_3, t_3) \wedge \mathsf{V}(v_1, t_1) \wedge \mathsf{V}(v_2, t_2) \wedge \mathsf{V}(v_3, t_3) \rightarrow \bot.$$

To find an appropriate value for p, consider a fixed hypothesis (which specifies an assignment for the existentially quantified variables), and a single database that contains exactly one of each of the pairs of tuples $\mathsf{V}(y_j, 0), \mathsf{V}(y_j, 1)$, for each y_j, $1 \leq j \leq m$. This complete assignment has a probability of $p^m(1-p)^m$ (if we ignore all other atoms).

We now compute the probability mass of all incomplete assignments for $y_1, \ldots, y_m$. For a fixed number k of "incomplete variables", there are $\binom{m}{k} 2^{m-k}$ such assignments, since we can first choose k out of m variables y_j for which neither atom $\mathsf{V}(y_j, 0), \mathsf{V}(y_j, 1)$ is true, and we have binary choice for each of the remaining $m - k$ variables. Each such assignment has a probability of $p^{m-k}(1-p)^{m+k}$, and hence in total the incomplete assignments have a probability of

$$\begin{aligned}\sum_{k=1}^{m} \tbinom{m}{k} 2^{m-k} p^{m-k} (1-p)^{m+k} &= 2^m p^m (1-p)^m \sum_{k=1}^{m} \tbinom{m}{k} \left(\tfrac{1-p}{2p}\right)^k \\ &= 2^m p^m (1-p)^m \left(\left(1 + \tfrac{1-p}{2p}\right)^m - 1\right)\end{aligned}$$

by the binomial theorem. Recall that our goal is to make this number smaller than $p^m(1-p)^m$, and hence we need to solve the inequation

$$1 > \left(\tfrac{p+1}{p}\right)^m - 2^m,$$

which is equivalent to

$$p > \frac{1}{(1+2^m)^{\frac{1}{m}} - 1}.$$

Since the latter term is always smaller than 1, it is possible to choose p as required, e.g., $p = 1 - 2^{-2m}$, which has only linearly many digits.

Now we have that Φ is valid if and only if there exists a hypothesis $\mathcal{H} \models (Q_\Phi, \Sigma)$ whose consistent extensions sum up to probability (under $\mathcal{P}_\Phi$) of at least $0.25^n \cdot c \cdot p^m(1-p)^m$.

Indeed, if there exists such an $\mathcal{H}$, then it represents a truth assignment for $x_1, \ldots, x_m$, which accounts for the term 0.25^n (see the proof of Theorem 7.33). But then to obtain the probability threshold, there must exist at least c complete assignments for $y_1, \ldots, y_m$, since the incomplete assignments on their own do not add up to $p^m(1-p)^m$. Conversely, if Φ is valid, we can use that information to choose a hypothesis that admits at least c extensions that represent complete truth assignments for $y_1, \ldots, y_m$. Observe that the program used in the reduction does not depend on the input formula and all the arities of the predicated are bounded, which lets us to conclude that $\mathsf{MPH}(\mathsf{UCQ}, \mathsf{NC})$ is $\mathrm{NP}^{\mathrm{PP}}$-hard in both fixed-program and bounded-arity complexity. □

Our results entail an interesting connection to the data complexity dichotomy for probabilistic query evaluation in PDBs (Dalvi and Suciu 2012). We build on Theorem 7.32 and show a direct reduction from MPH for FO-rewritable languages to probabilistic query evaluation in PDBs, which implies a data complexity dichotomy between P and PP (Dalvi and Suciu 2012).

Theorem 7.35 (dichotomy) *For FO-rewritable languages $\mathcal{L}$, $\mathsf{MPH}(\mathsf{UCQ}, \mathcal{L})$ is either in* P *or* PP*-hard in data complexity under polynomial time Turing reductions.*

Proof. Recall the proof of Theorem 7.32. According to (Dalvi and Suciu 2012), the evaluation problem for the UCQ $Q_{\bot,\Sigma}$ over a PDB $\mathcal{P}$ is either in P or PP-hard (under polynomial time Turing reductions). In the former case, MPH can also be decided in deterministic polynomial time. In the latter case, we reduce the evaluation problem for $Q_{\bot,\Sigma}$ over a PDB $\mathcal{P}$ to the MPH for Q_Σ and $Q_{\bot,\Sigma}$ over some PDB $\hat{\mathcal{P}} \supseteq \mathcal{P}$.

For the reduction, we introduce an "artificial match" for Q_Σ into $\hat{\mathcal{P}}$, by adding new constants and atoms (with probability 1) that satisfy one disjunct of Q_Σ, while taking care that these new atoms do not satisfy $Q_{\bot,\Sigma}$. Such atoms must exist if Q_Σ is not subsumed by $Q_{\bot,\Sigma}$; otherwise, all hypotheses would trivially have the probability 0 (and hence the MPH would be decidable in polynomial time). In $\hat{\mathcal{P}}$, the probability of the most probable hypothesis for Q_Σ and $Q_{\bot,\Sigma}$ is the same as the probability of $Q_{\bot,\Sigma}$ over $\mathcal{P}$, and hence deciding the threshold is PP-hard by (Dalvi and Suciu 2012) and Corollary 3.18. □

With this dichotomy result, we conclude our complexity analysis regarding MPH for OMQs. We now provide an overview of the results.

Overview of the Results

The complexity results for MPH are summarized in Table 7.6. As for MPD, all the deterministic complexity classes from Table 7.6 are a consequence of Theorem 7.30 which is of a generic nature. We discuss the remaining results in more detail.

Observe that the data complexity results for MPH are not as uniform as MPD. In Theorem 7.32, we have shown a nontrivial PP upper bound for MPH in data complexity. This result is further strengthened by a classification result given in Theorem 7.35, which asserts that MPH is either in polynomial time or PP-hard for OMQs relative to FO-rewritable Datalog$^{\pm}$ languages. In contrast, by Theorem 7.33, MPH is $\mathrm{NP}^{\mathrm{PP}}$-complete for Datalog$^{\pm}$ classes, for which OMQA is P-complete in data complexity.

Table 7.6: Complexity results for MPH for ontology-mediated queries. Results marked with * hold under poynomial time Turing reductions; all the remaining results are shown under standard many-one reductions.

Datalog$^{\pm}$ Languages	**Most Probable Hypothesis**			
	data	fixed-program	bounded-arity	combined
NC	in P vs PP*	NP^{PP}	NP^{PP}	NP^{PP}
L LF, AF	in P vs PP*	NP^{PP}	NP^{PP}	PSPACE
S, SF	in P vs PP*	NP^{PP}	NP^{PP}	EXP
A	in P vs PP*	NP^{PP}	P^{NE}	P^{NE}
GF, F	NP^{PP}	NP^{PP}	NP^{PP}	EXP
G	NP^{PP}	NP^{PP}	EXP	2EXP
WS, WA	NP^{PP}	NP^{PP}	2EXP	2EXP
WG	EXP	EXP	EXP	2EXP

This contrast in data complexity disappears for the fixed-program complexity as already MPH(UCQ, NC) is NP^{PP}-complete in fixed-program complexity and also in bounded-arity complexity (see Theorem 7.34). Observe also that all remaining lower bounds in bounded-arity and combined complexity follow from Theorem 7.31, which shows a direct reduction from MPD to MPH in all cases except for the data complexity.

7.4 Related Work and Outlook

Our work builds on the research on probabilistic databases (Imieliński and Lipski 1984; Suciu et al. 2011), which we already reviewed as part of the state of the art in Chapter 3. Our focus is on tuple-independent probabilistic databases, with an emphasis on the dichotomy result of (Dalvi and Suciu 2012). We first review related work for probabilistic OMQ evaluation and afterwards for the problems of most probable world and most probable hypothesis for OMQs.

Ontology-Mediated Queries and (Open-World) Probabilistic Databases. Motivated by the need of incorporating commonsense knowledge, we extend PDBs (and OpenPDBs) with ontological knowledge based on the possible world semantics. In the resulting formalism, ontological rules induce dependencies and the tuple-independence assumption of PDBs is relaxed as a consequence of such rules. We note that incorporating rules to reason over probabilistic facts is an old idea (Poole 1997; Sato 1995). The main difference of our approach is in the semantics of the underlying language.

Our framework enriches PDBs and OpenPDBs further by mediating the query with an ontology, where the query evaluation problem over a database is replaced with a *logical entailment* problem, allowing us to deduce implicitly encoded facts. Interpreting databases under commonsense knowledge in the form of ontologies is a well-known paradigm (Poggi et al. 2008), which has been very widely studied in the context of classical databases.

Ontology-based access to probabilistic data has been studied before for lightweight Description Logics, and the data complexity dichotomy result of PDBs is lifted to the light-weight description logics $\mathcal{EL}$ and *DL-Lite* over PDBs (Jung and Lutz 2012), whereby authors also describe the case of an ontology language that is not FO-rewritable and causes all CQs of a certain form to become #P-hard. Note that this approach allows only a *unique probability distribution*, and it assigns the probability zero to facts that are not entailed by the knowledge base; therefore, it is very different than probabilistic OMQ evaluation in OpenPDBs. We consider this as a very closely related work to probabilistic OMQ evaluation in PDBs, but note that we consider the more expressive languages of the Datalog$^{\pm}$ family and we also provide combined complexity results (all of which are of a decision-theoretic nature).

Recall that both PDBs and OpenPDBs assume a fixed and finite domain. This is also relaxed in the semantics of Datalog$^{\pm}$. More precisely, note that Datalog$^{\pm}$ vocabulary additionally allows infinitely many *nulls*. Therefore, although the probability distribution is still defined over finitely many objects, we can now reason over domains, which include an infinite number of unknown objects. This is also a main difference with (function-free) probabilistic logic programming (De Raedt, Kimmig, and Toivonen 2007) and Markov Logic Networks (Richardson and Domingos 2006) as already pointed out in the introduction.

Another probabilistic extensions of Datalog$^{\pm}$ is given in (Gottlob, Lukasiewicz, Martinez, and Simari 2013), which result from a combination of ontologies and Markov logic networks (Richardson and Domingos 2006). Both the semantics and the assumptions used in this works is very different than ours. Rather less closely related work is probabilistic XML (Abiteboul, Chan, Kharlamov, Nutt, and Senellart 2011; Kimelfeld and Senellart 2013). We will review other probabilistic ontology languages based on Description Logics in Chapter 8.

There are several interesting directions of future work. We clearly need to develop systems that can do probabilistic reasoning on a large-scale, which is a very challenging problem and a long-term goal. Nevertheless, the technical results presented in this work equips us with some concrete directions to achieve this goal. As a first step, we can use a query rewriting tool for ontology-mediated query answering and reduce probabilistic OMQ evaluation to query evaluation in probabilistic relational databases. Besides, for all problems, studied in this chapter, we have presented a mostly complete picture in terms of the computational complexity of these problems. There are, however, many interesting problems left open. For instance, it is open whether a data complexity dichotomy result between P and PP holds for $\mathsf{PQE}(\mathsf{UCQ}, \mathsf{G})$ in PDBs.

Finally, we note that the tuple-independence assumption is not a semantic baseline and rather a requirement in the computational sense. As pointed out before, ontological rules break down the independence of facts and improves some query answers. However, all the remaining facts are still assumed to be independent. Besides, for probabilistic OMQ evaluation in OpenPDBs, we focused on only consistent completions. It is more reasonable to define an entropy principle so that the inconsistent completions ruled out in the semantics. A promising direction is the random worlds semantics (Grove, J. Y. Halpern, and Koller 1994), which we also leave as future work.

Ontology-Mediated Queries and Maximal Posterior Computations. Our work is inspired by the *maximal posterior probability* computations in PGMs (Koller and Friedman 2009; Pearl 1988). It is well-known that they are a form of *abduction*, which clearly also applies to the problems studied here. Maximal posterior probability computations are central in PGMs and in statistical relational learning. However, analogous problems have not been studied in depth in the context of PDBs. To our knowledge, the only work in this direction is on most probable databases (Gribkoff, Van den Broeck, and Suciu 2014a), which we use as a starting point.

Most of our data complexity results are covered by the classes NP, Σ_2^{P}, PP, and $\mathrm{NP}^{\mathrm{PP}}$. Though intractable, they are at the core of many important problems, which motivated a body of work tailored towards scalable algorithms for these classes. There is an immediate connection between MPE and weighted MAX-SAT (Sang, Beame, and Kautz 2007). Similarly, advances in knowledge compilation (Park and Darwiche 2004a; Pipatsrisawat and Darwiche 2009) and model counting (Chakraborty, Meel, and Vardi 2016; Fremont, Rabe, and Seshia 2017) are tailored to achieve optimal, scalable algorithms for problems related to classes such as PP and $\mathrm{NP}^{\mathrm{PP}}$.

Finally, both MPD and MPH can be cast into their propositional variants using the lineage representation of queries, which gives immediate access to such algorithms. However, there is need for future work in this direction, as grounding is not an optimal way of handling these problems; performing inference directly on FO-structures is known to be more efficient.

Chapter 8

Probabilistic Data Access with Description Logics

One of the central limitations of Description Logics, in their classical form, is their inability to handle uncertain knowledge and data. Since uncertainty is inherent to many application domains and data collection systems (such as large-scale probabilistic knowledge bases), it is fundamental for DLs to provide effective methods to handle it.

One recent proposal for dealing with uncertain knowledge is in the form of Bayesian ontology languages (Ceylan and Peñaloza 2014b, 2017). The basic idea behind these languages is to extend classical DL-based formalisms with a compact representation of the joint probability of the axioms in the ontology with the help of a Bayesian network. Compared to other approaches for dealing with uncertainty, Bayesian ontology languages have the advantage of making no implicit independence assumptions, while still remaining succinct.

On the one hand, Bayesian Ontology Languages extend classical ontology languages with the capability of representing and reasoning over uncertain domains. On the other hand, one can think of Bayesian ontology languages as a generalization of BNs. Under this view, every valuation of the variables in the BN corresponds to a classical *first-order* KB, rather than just a propositional world.

We first study ontology-mediated query answering in Bayesian ontology languages, called Bayesian query evaluation and provide a thorough investigation. In the second part of this chapter, we propose a novel monitoring approach that combines the power of Bayesian ontology languages with dynamic BNs. The resulting formalism is then called dynamic Bayesian ontology languages (DBOLs). Briefly, we use dynamic BNs to define a state-transition system over a distinguished and fixed set of variables to be monitored, and ontologies to encode the rich background knowledge.

8.1 Bayesian Ontology Languages

Bayesian ontology languages (BOLs) are introduced as a means of modeling structured knowledge in uncertain domains (Ceylan and Peñaloza 2014b, 2017). The main assumption in BOLs is that every piece of ontological knowledge is required to hold in a given context, which is formalized as a propositional formula over the variables of a BN. By specifying a probability distribution over the contexts (through the BN), BOLs allow for ontological reasoning under uncertainty. We shortly revisit the basics of BOLs and study ontology-mediated query answering for BOLs.

The vocabulary of Bayesian ontology languages extends classical DL vocabulary with the variables V of the given BN. Bayesian ontology languages extend the syntax of the underlying ontology language using propositional annotations, called contexts.

Definition 8.1 Let $\mathcal{B}$ be a BN defined over the variables V. A *context over a BN* $\mathcal{B}$ is a propositional formula φ over the variables V of $\mathcal{B}$. A *probabilistic axiom (over a BN* $\mathcal{B}$*)* is of the form $\langle \alpha : \varphi \rangle$, where α is a TBox axiom, and φ is a context over $\mathcal{B}$. Similarly, a *probabilistic assertion (over a BN* $\mathcal{B}$*)* is of the form $\langle \alpha : \varphi \rangle$, where α is an assertion, and φ is a context over $\mathcal{B}$. ◊

In the following, we write α to abbreviate $\langle \alpha : \mathbf{true} \rangle$, where **true** denotes the context that is satisfied by all the worlds, also called the global context. Given the definitions of probabilistic axioms/assertions, probabilistic knowledge bases are defined as follows.

Definition 8.2 A *probabilistic TBox* $\mathcal{T}$ *(over a BN* $\mathcal{B}$*)* is a finite set of probabilistic axioms over to a BN $\mathcal{B}$. A *probabilistic ABox* $\mathcal{A}$ *(over a BN* $\mathcal{B}$*)* is a finite set of probabilistic assertions over a BN $\mathcal{B}$. A *probabilistic knowledge base* is a triple $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$, where $\mathcal{B}$ is a BN, $\mathcal{T}$ is a probabilistic TBox over $\mathcal{B}$, and $\mathcal{A}$ is a probabilistic ABox over $\mathcal{B}$. ◊

Whenever the BN is clear from the context, we will simply write (probabilistic) TBox/ABox, and (probabilistic) axiom/assertion, for ease of presentation.

Semantically, BOLs are composed of two layers as we discuss next. The following definition introduces interpretations, which extend classical DL interpretations to be able to decide when a probabilistic axiom or assertion is satisfied.

Definition 8.3 An *interpretation* is a tuple $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \mathcal{V}^{\mathcal{I}})$ where $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a classical first-order interpretation, and $\mathcal{V}^{\mathcal{I}}$ is a valuation of the variables of the BN. The interpretation $\mathcal{I}$ *satisfies* a probabilistic axiom (or a probabilistic assertion) of the form $\langle \alpha : \varphi \rangle$, denoted $\mathcal{I} \models \langle \alpha : \varphi \rangle$, if and only if either (i) $\mathcal{V}^{\mathcal{I}} \not\models \varphi$, or (ii) $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}) \models \alpha$. The interpretation $\mathcal{I}$ is a *model of a probabilistic TBox* $\mathcal{T}$ *(resp., ABox* $\mathcal{A}$*)* if it satisfies all probabilistic axioms (resp., probabilistic assertions) in $\mathcal{T}$ (resp., $\mathcal{A}$). ◊

The second layer of the semantics introduces probabilistic interpretations, which define a probability distribution over a set of interpretations. To be a model, a probabilistic interpretation is required to define a probability distribution that is consistent with the BN.

Definition 8.4 A *probabilistic interpretation* is a pair $\mathbf{I} = (\mathfrak{I}, \mathrm{P}_{\mathfrak{I}})$, where $\mathfrak{I}$ is a set of interpretations $\mathcal{I}$, and $\mathrm{P}_{\mathfrak{I}}$ is a probability distribution over $\mathfrak{I}$ such that $\mathrm{P}_{\mathfrak{I}}(\mathcal{I}) > 0$ only for finitely many interpretations $\mathcal{I} \in \mathfrak{I}$. $\mathbf{I}$ is a *model* of the probabilistic TBox $\mathcal{T}$ (resp., ABox $\mathcal{A}$) if every $\mathcal{I} \in \mathfrak{I}$ is a model of $\mathcal{T}$ (resp., $\mathcal{A}$). $\mathbf{I}$ is *consistent* with the BN $\mathcal{B}$ if for every valuation $\mathcal{W}$ of V it holds that

$$\sum_{\mathcal{I} \in \mathfrak{I}, \mathcal{V}^{\mathcal{I}} = \mathcal{W}} \mathrm{P}_{\mathfrak{I}}(\mathcal{I}) = \mathrm{P}_{\mathcal{B}}(\mathcal{W}).$$

The probabilistic interpretation $\mathbf{I}$ is a *model* of the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ if and only if it is a model of $(\mathcal{T}, \mathcal{A})$, and it is consistent with $\mathcal{B}$. ◊

Consider again the health-care scenario encoded into a BN as given in Figure 8.1. Suppose that we would like to encode the global knowledge about patients, such as *high systolic pressure* implies *hypertension*, or *any male patient with a finding of hypertension* is under the risk of *having myocardial infarction.* These are typical statements of a *first-order* nature, which is in widespread use in health-care scenarios (Pisanelli 2004),

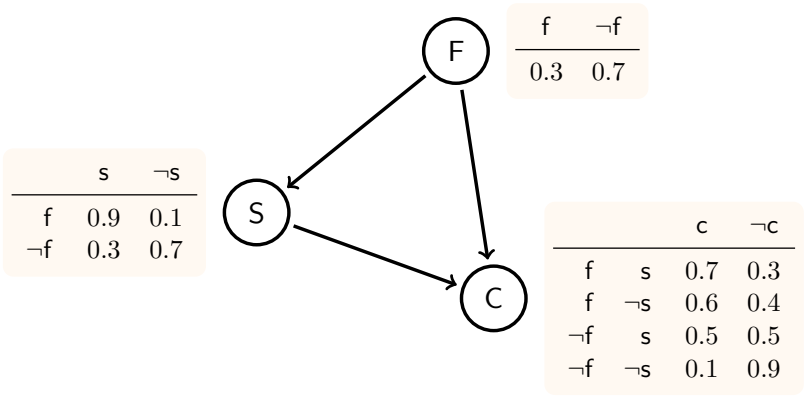


Figure 8.1: The BN $\mathcal{B}_h$ over the variables $V_h = \{\mathsf{F}, \mathsf{S}, \mathsf{C}\}$ (repeated from Figure 5.1).

and can be conveniently modeled by choosing an appropriate ontology language. The following example shows that BOLs are appropriate for modeling such scenarios.

Example 8.5 Consider the TBox $\mathcal{T}_h$ that consists of the axioms

$$\exists \mathsf{pressure.High} \equiv \exists \mathsf{finding.Hypertension},$$
$$\exists \mathsf{cholesterol.High} \equiv \exists \mathsf{finding.Hypercholesteremia},$$
$$\exists \mathsf{finding.Hypertension} \sqcap \exists \mathsf{finding.Hypercholesteremia} \sqcap \mathsf{Male} \sqsubseteq \exists \mathsf{risk.MyocardialInfarction},$$
$$\mathsf{MyocardialInfarction} \sqsubseteq \mathsf{CriticalSituation}.$$

Note that the TBox is a classical one, i.e., it uses only axioms with global context. For instance, the first axiom in the TBox states that anyone with a high systolic pressure has a finding of hypertension. Consider further the ABox $\mathcal{A}_h$, which contains the assertions

$$\mathsf{Patient(bob)}, \mathsf{Male(bob)}, \mathsf{High(h)},$$
$$\langle \mathsf{pressure(bob, h)} : \mathsf{s} \rangle, \langle \neg \mathsf{pressure(bob, h)} : \neg \mathsf{s} \rangle,$$
$$\langle \mathsf{Fatigue(bob)} : \mathsf{f} \rangle, \langle \mathsf{cholesterol(bob, h)} : \mathsf{c} \rangle, \langle \neg \mathsf{cholesterol(bob, h)} : \neg \mathsf{c} \rangle.$$

The ABox $\mathcal{A}_h$ contains probabilistic assertions, such as $\langle \mathsf{cholesterol(bob, h)} : \mathsf{c} \rangle$, and therefore, it is a probabilistic ABox. The probabilistic information is associated with the BN $\mathcal{B}_h$, depicted in Figure 8.1. Then, $\mathcal{K}_h = (\mathcal{T}_h, \mathcal{A}_h, \mathcal{B}_h)$ defines a probabilistic KB. $\Diamond$

It is easy to see that Bayesian ontology languages are a generalization of BNs. Furthermore, they are also a generalization of ontology languages, i.e., if we associate all axioms and assertions with the global context, we obtain a classical knowledge base. Importantly, with the help of ontologies, we gain additional expressivity, and we can encode structured information in a more concise and adequate manner. To date, the focus on BOLs was on *light-weight* ontology languages and on classical reasoning tasks such as *subsumption* (Ceylan and Peñaloza 2014b). We study ontology-mediated query answering for BOLs and consider different, prominent DLs as the underlying ontology languages.

8.1.1 Query Evaluation in Bayesian Ontology Languages

Querying ontologies is an important task, and conjunctive queries form the central querying tool for classical ontologies. On the other hand, inference in BNs is based on primitive queries in the form of propositional events, and answers to such queries are probabilistic. Being a combination of these formalisms, Bayesian ontology languages provide both a rich query language and probabilistic answers to queries.

Definition 8.6 (semantics) Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ be a probabilistic KB, and Q be a Boolean query. The probability of Q w.r.t. a probabilistic interpretation $\mathbf{I} = (\mathfrak{I}, \mathrm{P}_{\mathfrak{I}})$, denoted $\mathrm{P}_{\mathbf{I}}(Q)$, is given as follows:

$$\mathrm{P}_{\mathbf{I}}(Q) \coloneqq \sum_{\mathcal{I} \in \mathfrak{I}, \mathcal{I} \models Q} \mathrm{P}_{\mathfrak{I}}(\mathcal{I}).$$

The *lower* (resp., *upper*) *probability* of a Boolean query Q, denoted $\underline{\mathrm{P}}_{\mathcal{K}}(Q)$ (resp., $\overline{\mathrm{P}}_{\mathcal{K}}(Q)$) w.r.t. $\mathcal{K}$ is given as follows:

$$\underline{\mathrm{P}}_{\mathcal{K}}(Q) \coloneqq \inf_{\mathbf{I} \models \mathcal{K}} \mathrm{P}_{\mathbf{I}}(Q) \qquad (\text{resp., } \overline{\mathrm{P}}_{\mathcal{K}}(Q) \coloneqq \sup_{\mathbf{I} \models \mathcal{K}} \mathrm{P}_{\mathcal{K}}(Q)).$$ ◊

In a nutshell, every Boolean query is now associated with a lower and upper probability value, which draw boundaries to the likelihood of the query. Intuitively, the lower probability is determined based on the *given* knowledge that is currently known, and the upper probability is determined based on *consistent completions* of the given knowledge. This distinction is important both in theoretical and in practical sense.

From a theoretical perspective, ontologies do not make any completeness assumptions. Therefore, limiting the probabilistic model to only consider lower probabilities, would also limit the query semantics to the knowledge that is known. This would not be in line with the nature of these formalisms. From a practical perspective, while querying highly incomplete sources, lower probabilities will not be very informative for many queries, i.e., they will evaluate to very low probabilities, making them hardly distinguishable. In such cases, upper probabilities can provide meaningful answers on how possible, or impossible such queries are.

Our choice for lower and upper probabilities can be seen analogous to *credal networks* (Cozman 2000), which generalize Bayesian networks with sets of probability distributions. Nevertheless, in our framework, we still require a unique probability distribution to be specified, which, in turn, restricts the set of all probability distributions through probabilistic interpretations.

Definition 8.7 (Bayesian query evaluation) Given a probabilistic knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$, a UCQ Q, and a probability $p \in (0, 1]$, *lower* (resp., *upper*) *Bayesian query evaluation*, denoted $\underline{\mathsf{B}}\mathsf{QE}$ (resp., $\overline{\mathsf{B}}\mathsf{QE}$), is to decide whether $\underline{\mathrm{P}}_{\mathcal{K}}(Q) > p$ (resp., $\overline{\mathrm{P}}_{\mathcal{K}}(Q) > p$). $\underline{\mathsf{B}}\mathsf{QE}$ (resp., $\overline{\mathsf{B}}\mathsf{QE}$) is parametrized with the language of the ontology and the query; we write $\underline{\mathsf{B}}\mathsf{QE}(\mathsf{Q}, \mathcal{L})$ (resp., $\overline{\mathsf{B}}\mathsf{QE}(\mathsf{Q}, \mathcal{L})$) to define BQE on the class Q of queries and on the class of ontologies restricted to the language $\mathcal{L}$. ◊

Note that we ignore *inconsistencies*, as before. This is a reasonable assumption for this problem, since it is always possible to normalize the probability distribution

to only consider the consistent worlds. On the other hand, there are more elaborate methods, known as *inconsistency-tolerant reasoning*, in database theory (Arenas, Bertossi, and Chomicki 1999), as well as in DLs (Lembo, Lenzerini, Rosati, Ruzzi, and Savo 2010), which recently has also been investigated in the context of probabilistic ontology languages (Ceylan, Lukasiewicz, and Peñaloza 2016). These are outside the focus of the current work.

We first show a naïve approach for computing the lower probability of a query. Formally, for a valuation $\mathcal{W}$ of the variables V of the BN, we define $\mathcal{K}_\mathcal{W} = (\mathcal{T}_\mathcal{W}, \mathcal{A}_\mathcal{W})$, where

$$\mathcal{T}_\mathcal{W} := \{\alpha \mid \langle \alpha : \varphi \rangle \in \mathcal{T}, \mathcal{W} \models \varphi\} \quad \text{and} \quad \mathcal{A}_\mathcal{W} := \{\alpha \mid \langle \alpha : \varphi \rangle \in \mathcal{A}, \mathcal{W} \models \varphi\}.$$

Intuitively, $\mathcal{K}_\mathcal{W}$ represents an abstraction of the probabilistic knowledge base $\mathcal{K}$ with respect to a valuation $\mathcal{W}$. Thus, for every world $\mathcal{W}$, $\mathcal{K}_\mathcal{W}$ defines a classical knowledge base containing *all and only those axioms* that are required to hold in this world. Based on this, we are able to compute the probability of a query by naively adding the probability $\mathrm{P}_\mathcal{B}(\mathcal{W})$ for every world $\mathcal{W}$ that satisfies $\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q)$, as summarized next.

Theorem 8.8 *Given a probabilistic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ and a Boolean query Q, it holds that*

$$\underline{\mathrm{P}}_\mathcal{K}(Q) = \sum_{\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q)} \mathrm{P}_\mathcal{B}(\mathcal{W}),$$

where $\mathcal{W}$ denotes a valuation over the variables of the BN $\mathcal{B}$, $\mathcal{A}_\mathcal{W}$ is a classical ABox, $\mathcal{T}_\mathcal{W}$ is a classical TBox and $\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q)$ is classical OMQA.

Proof. For the given probabilistic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ and the query Q, we define a probabilistic interpretation $\mathbf{I} = (\mathfrak{I}, \mathrm{P}_\mathfrak{I})$ where $\mathfrak{I}$ contains, for every valuation $\mathcal{W}$ of the variables V, exactly one interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I}, \mathcal{V}^\mathcal{I})$ such that

$$\begin{aligned} \mathcal{V}^\mathcal{I} &= \mathcal{W}, \\ (\Delta^\mathcal{I}, \cdot^\mathcal{I}) &\models (\mathcal{T}_\mathcal{W}, \mathcal{A}_\mathcal{W}), \\ (\Delta^\mathcal{I}, \cdot^\mathcal{I}) &\models Q \ \text{ if and only if } \ \mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q). \end{aligned} \tag{8.1}$$

Notice that the existence of a model for $(\mathcal{T}_\mathcal{W}, \mathcal{A}_\mathcal{W})$ is ensured by our consistency assumption. Similarly, the last condition can always be fulfilled, i.e., if $\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q)$, then we can choose any model, otherwise, we choose one model such that $(\Delta^\mathcal{I}, \cdot^\mathcal{I}) \not\models Q$, which must exist by definition. Furthermore, $\mathrm{P}_\mathfrak{I}$ defines a probability distribution over the interpretations $\mathcal{I} \in \mathfrak{I}$ such that

$$\mathrm{P}_\mathfrak{I}(\mathcal{I}) = \mathrm{P}_\mathcal{B}(\mathcal{W}) \text{ if and only if } \mathcal{V}^\mathcal{I} = \mathcal{W}. \tag{8.2}$$

We show that $\mathbf{I}$ is a model of $\mathcal{K}$. We start by showing that $\mathcal{I}$ is a model of $\mathcal{T}$ and of $\mathcal{A}$ for all $\mathcal{I} \in \mathfrak{I}$. Observe that, for all axioms $\langle \alpha : \varphi \rangle \in \mathcal{T}$, if the valuation $\mathcal{W}$ satisfies φ, it follows that $\mathcal{I} \models \langle \alpha : \varphi \rangle$ since $(\Delta^\mathcal{I}, \cdot^\mathcal{I})$ is a model of $\mathcal{T}_\mathcal{W}$ by (8.1). If, on the other hand, the valuation $\mathcal{W}$ does not satisfy φ for some axiom $\langle \alpha : \varphi \rangle \in \mathcal{T}$ then $\mathcal{I} \models \langle \alpha : \varphi \rangle$ is a trivial consequence of the semantics. Thus, we have shown that $\mathcal{I}$ is a model of $\mathcal{T}$. By analogous arguments, it is easy to see that $\mathcal{I}$ is also a model of $\mathcal{A}$. Moreover, the probability distribution $\mathrm{P}_\mathfrak{I}$ is clearly consistent with the BN $\mathcal{B}$ by 8.2. Thus, we conclude that $\mathbf{I} \models \mathcal{K}$.

Finally, since $\mathcal{I} \models Q$ if and only if $\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q)$ for all $\mathcal{I} \in \mathfrak{I}$, where $\mathcal{V}^\mathcal{I} = \mathcal{W}$, we obtain that

$$\begin{aligned}\underline{\mathrm{P}}_\mathcal{K}(Q) = \inf_{\mathbf{I} \models \mathcal{K}} \mathrm{P}_\mathbf{I}(Q) \leq \mathrm{P}_\mathbf{I}(Q) &= \sum_{\mathcal{I} \in \mathfrak{I}, \mathcal{I} \models Q} \mathrm{P}_\mathfrak{I}(\mathcal{I}) \\ &= \sum_{\substack{\mathcal{V}^\mathcal{I} = \mathcal{W} \\ \mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q)}} \mathrm{P}_\mathfrak{I}(\mathcal{I}) \\ &= \sum_{\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q)} \mathrm{P}_\mathcal{B}(\mathcal{W}).\end{aligned}$$

Suppose now that the inequality is strict. Then, there must exist a model $\mathcal{H} = (\mathfrak{J}, \mathrm{P}_\mathfrak{J})$ such that

$$\mathrm{P}_\mathcal{H}(Q) < \sum_{\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q)} \mathrm{P}_\mathcal{B}(\mathcal{W}).$$

Without loss of generality, we can assume that the probabilistic model $\mathcal{H}$ contains exactly one interpretation $\mathcal{I} \in \mathfrak{J}$ for every world $\mathcal{W}$. Such models, called pithy models, always exist (Ceylan and Peñaloza 2017). This yields

$$\mathrm{P}_\mathcal{H}(Q) = \sum_{\mathcal{I} \in \mathfrak{J}, \mathcal{I} \models Q} \mathrm{P}_\mathfrak{J}(\mathcal{I}) < \sum_{\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q)} \mathrm{P}_\mathcal{B}(\mathcal{W}),$$

which holds if and only if $\mathrm{P}_\mathfrak{J}(\mathcal{I}) < \mathrm{P}_\mathcal{B}(\mathcal{W})$ for some $\mathcal{I} \in \mathfrak{J}$ where $\mathcal{V}^\mathcal{I} = \mathcal{W}$. Since, $\mathcal{H}$ is a pithy model, $\mathcal{I}$ is the only interpretation with $\mathcal{V}^\mathcal{I} = \mathcal{W}$. This implies that the probability distribution $\mathrm{P}_\mathfrak{J}$ is not consistent with the BN $\mathcal{B}$, which contradicts the assumption that $\mathcal{H}$ is a model. □

To compute the upper probability of a query, we need to identify models that entail the query in as many worlds as possible. Since ontologies employ the OWA, it is always possible to define a model that entails a query, except for the case when explicit negative information is given about the query. Thus, it is sufficient to compute the probabilities of the worlds that contradict the query and then subtract the probability of such worlds from 1.

To be able to detect the worlds where the query can not be entailed (without implying an inconsistency), we transform the query into its dual form. Formally, given a Boolean query Q, the lower and upper probabilities for the negated query $\neg Q$ are defined as before. Then, we show that computing the upper probability of a query Q can be reduced to computing the lower probability of $\neg Q$.

Theorem 8.9 *Given a probabilistic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ and a* UCQ *Q, it holds that*

$$\overline{\mathrm{P}}_\mathcal{K}(Q) = 1 - \underline{\mathrm{P}}_\mathcal{K}(\neg Q).$$

Proof. Given a probabilistic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ and a query $Q = Q_1 \vee \ldots \vee Q_n$, let $\mathsf{gr}(Q_i)$ be the set of atoms from a grounding of Q_i with fresh constants. Observe that

$\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, \neg Q)$ if and only if $\mathcal{A}_\mathcal{W} \cup \mathsf{gr}(Q_i) \models (\mathcal{T}_\mathcal{W}, \bot)$ for all $1 \leq i \leq n$, which implies that

$$\underline{\mathrm{P}}_\mathcal{K}(\neg Q) = \sum_{\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, \neg Q)} \mathrm{P}_\mathcal{B}(\mathcal{W}) = \sum_{\forall i: \mathcal{A}_\mathcal{W} \cup \mathsf{gr}(Q_i) \models (\mathcal{T}_\mathcal{W}, \bot)} \mathrm{P}_\mathcal{B}(\mathcal{W}).$$

Thus, to prove $\overline{\mathrm{P}}_\mathcal{K}(Q) = 1 - \underline{\mathrm{P}}_\mathcal{K}(\neg Q)$, it is sufficient to show that

$$\overline{\mathrm{P}}_\mathcal{K}(Q) = 1 - \sum_{\forall i: \mathcal{A}_\mathcal{W} \cup \mathsf{gr}(Q_i) \models (\mathcal{T}_\mathcal{W}, \bot)} \mathrm{P}_\mathcal{B}(\mathcal{W}). \tag{8.3}$$

To prove the inequality $\geq$ for the Equation 8.3, we define a probabilistic interpretation $\mathbf{I} = (\mathfrak{I}, \mathrm{P}_\mathfrak{I})$ where $\mathfrak{I}$ contains, for every valuation $\mathcal{W}$ of the variables V, exactly one interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I}, \mathcal{V}^\mathcal{I})$ such that

$$\begin{aligned} \mathcal{V}^\mathcal{I} &= \mathcal{W}, \\ (\Delta^\mathcal{I}, \cdot^\mathcal{I}) &\models (\mathcal{T}_\mathcal{W}, \mathcal{A}_\mathcal{W}), \\ (\Delta^\mathcal{I}, \cdot^\mathcal{I}) &\models Q \ \text{ iff } \ \mathcal{A}_\mathcal{W} \cup \mathsf{gr}(Q_i) \not\models (\mathcal{T}_\mathcal{W}, \bot) \text{ for some } 1 \leq i \leq n. \end{aligned} \tag{8.4}$$

As before, the existence of a model for $(\mathcal{T}_\mathcal{W}, \mathcal{A}_\mathcal{W})$ is ensured by our consistency assumption. Analogously, the last condition can always be fulfilled, i.e., if $\mathcal{A}_\mathcal{W} \cup \mathsf{gr}(Q_i) \not\models (\mathcal{T}_\mathcal{W}, \bot)$ then, we can find a model that satisfies Q_i, and thus Q. If, on the other hand, $\mathcal{A}_\mathcal{W} \cup \mathsf{gr}(Q_i) \models (\mathcal{T}_\mathcal{W}, \bot)$ then any model can be chosen since none of the queries $Q_1 \ldots Q_n$ is satisfiable by any of the models. Furthermore, $\mathrm{P}_\mathfrak{I}$ defines a probability distribution over the interpretations $\mathcal{I} \in \mathfrak{I}$ such that

$$\mathrm{P}_\mathfrak{I}(\mathcal{I}) = \mathrm{P}_\mathcal{B}(\mathcal{W}) \text{ if and only if } \mathcal{V}^\mathcal{I} = \mathcal{W}. \tag{8.5}$$

Based on Equivalence 8.4 and 8.5, we can apply analogous arguments as in Theorem 8.8 to show that $\mathbf{I}$ is indeed a model of $\mathcal{K}$. Then, we obtain that

$$\begin{aligned} \overline{\mathrm{P}}_\mathcal{K}(Q) = \sup_{\mathbf{I} \models \mathcal{K}} \mathrm{P}_\mathbf{I}(Q) \geq \mathrm{P}_\mathbf{I}(Q) = \sum_{\mathcal{I} \in \mathfrak{I}, \mathcal{I} \models Q} \mathrm{P}_\mathfrak{I}(\mathcal{I}) &= \sum_{\substack{\mathcal{V}^\mathcal{I} = \mathcal{W}, \\ \exists i: \mathcal{A}_\mathcal{W} \cup \mathsf{gr}(Q_i) \not\models (\mathcal{T}_\mathcal{W}, \bot)}} \mathrm{P}_\mathfrak{I}(\mathcal{I}) \\ &= 1 - \sum_{\substack{\mathcal{V}^\mathcal{I} = \mathcal{W}, \\ \forall i: \mathcal{A}_\mathcal{W} \cup \mathsf{gr}(Q_i) \models (\mathcal{T}_\mathcal{W}, \bot)}} \mathrm{P}_\mathfrak{I}(\mathcal{I}) \\ &= 1 - \sum_{\forall i: \mathcal{A}_\mathcal{W} \cup \mathsf{gr}(Q_i) \models (\mathcal{T}_\mathcal{W}, \bot)} \mathrm{P}_\mathcal{B}(\mathcal{W}). \end{aligned}$$

Suppose now that the inequality is strict. Then, there must exist a model $\mathbf{H} = (\mathfrak{J}, \mathrm{P}_\mathfrak{J})$ such that

$$\mathrm{P}_\mathbf{H}(Q) > 1 - \sum_{\forall i: \mathcal{A}_\mathcal{W} \cup \mathsf{gr}(Q_i) \models (\mathcal{T}_\mathcal{W}, \bot)} \mathrm{P}_\mathcal{B}(\mathcal{W}).$$

As before, we can assume that the probabilistic model $\mathbf{H}$ contains exactly one interpretation $\mathcal{I} \in \mathfrak{J}$ for every world $\mathcal{W}$ and get that

$$\mathrm{P}_\mathbf{H}(Q) = \sum_{\mathcal{I} \in \mathfrak{J}, \mathcal{I} \models Q} \mathrm{P}_\mathfrak{J}(\mathcal{I}) > 1 - \sum_{\forall i: \mathcal{A}_\mathcal{W} \cup \mathsf{gr}(Q_i) \models (\mathcal{T}_\mathcal{W}, \bot)} \mathrm{P}_\mathcal{B}(\mathcal{W}),$$

which holds if and only if it holds that

$$\sum_{\mathcal{I}\in\mathfrak{J},\mathcal{I}\models Q} \mathrm{P}_{\mathfrak{J}}(\mathcal{I}) > \sum_{\exists i:\mathcal{A}_{\mathcal{W}}\cup\mathsf{gr}(Q_i)\not\models(\mathcal{T}_{\mathcal{W}},\bot)} \mathrm{P}_{\mathcal{B}}(\mathcal{W}).$$

Since, **H** is a pithy model, we have that $\mathrm{P}_{\mathfrak{J}}(\mathcal{I}) = \mathrm{P}_{\mathcal{B}}(\mathcal{W})$ for all $\mathcal{I} \in \mathfrak{J}$ where $\mathcal{V}^{\mathcal{I}} = \mathcal{W}$. Then, there must exists an interpretation $\mathcal{I} \in \mathfrak{J}$ where $\mathcal{I} \models Q$ holds, while $\mathcal{A}_{\mathcal{W}} \cup \mathsf{gr}(Q_i) \models (\mathcal{T}_{\mathcal{W}}, \bot)$ for all i. The latter holds if and only if $\mathcal{A}_{\mathcal{W}} \models (\mathcal{T}_{\mathcal{W}}, \neg Q_i)$ for all i. Since **H** is a model, $\mathcal{I}$ must satisfy $(\mathcal{T}_{\mathcal{W}}, \mathcal{A}_{\mathcal{W}})$, by definition. But then, we obtain $\mathcal{I} \models \neg Q_i$ for all i, which implies that $\mathcal{I} \models \neg Q$, which is a contradiction. Thus, we have proven the Equivalence 8.3 and thus the result. □

With the help of Theorem 8.8 and Theorem 8.9, we can compute the lower and upper probabilities for ontological queries. We now illustrate Bayesian query answering on our running example and give more insights on these probability computations.

Example 8.10 Consider again the KB $\mathcal{K} = (\mathcal{T}_h, \mathcal{A}_h, \mathcal{B}_h)$. Instead of querying the BN directly thorough elementary probabilistic events, we can pose ontology-mediated queries. For instance, instead of querying whether a patient has high systolic pressure, we can now pose high-level queries, such as whether a patient is in a critical situation, specified by the query

$$Q_{\mathsf{bob}} = \exists x\ \mathsf{Patient}(\mathsf{bob}) \wedge \mathsf{risk}(\mathsf{bob}, x) \wedge \mathsf{CriticalSituation}(x).$$

It is easy to see that bob is a patient that is at risk of a critical situation, and we obtain

$$\underline{\mathrm{P}}_{\mathcal{K}_h}(Q_{\mathsf{bob}}) = \sum_{\mathcal{A}_{\mathcal{W}}\models(\mathcal{T}_{\mathcal{W}},Q)} \mathrm{P}_{\mathcal{B}_h}(\mathcal{W}) = \mathrm{P}_{\mathcal{B}_h}(\mathsf{s} \wedge \mathsf{c}) = .294.$$

Furthermore, we also have $\overline{\mathrm{P}}_{\mathcal{K}_h}(Q_{\mathsf{bob}}) = 1$, as we can assert that bob is in a critical situation by adding the assertion $\mathsf{CriticalSituation}(\mathsf{bob})$, and this is not in conflict with existing knowledge. In fact, there may be other types of critical situations beyond the ones described by the ontology. Consider, on the other hand, the query

$$Q'_{\mathsf{bob}} = \exists x\ \mathsf{finding}(\mathsf{bob}, x) \wedge \mathsf{Hypertension}(x).$$

For Q'_{bob}, we observe that

$$\underline{\mathrm{P}}_{\mathcal{K}_h}(Q'_{\mathsf{bob}}) = \mathrm{P}_{\mathcal{B}_h}(\mathsf{s}) = \overline{\mathrm{P}}_{\mathcal{K}_h}(Q'_{\mathsf{bob}}) = .48,$$

since bob can have hypertension if and only if he has high systolic pressure. Thus, we exclude all worlds where bob does not have a high systolic pressure. ◊

As pointed before, the lower probability is determined based on the given knowledge, and the upper probability is determined based on consistent completions of the given knowledge. For instance, the knowledge base $\mathcal{K}_h$ asserts that high fever can cause fatigue, but does not exclude the possibility of other causes of fatigue, even though nothing specific is known about this.

Complexity Results for Bayesian Query Evaluation

In our technical analysis, we study *data complexity* and *combined complexity* for probabilistic Bayesian query evaluation, as is customary in DLs. We first discuss $\underline{\mathsf{B}}\mathsf{QE}$ and then extend our analysis $\overline{\mathsf{B}}\mathsf{QE}$.

Complexity Results for $\underline{\mathsf{B}}\mathsf{QE}$

We first show how Theorem 8.8 can be exploited in order to obtain rather general membership results for lower Bayesian query answering.

Theorem 8.11 *Let $\mathfrak{C}$ denote the data (respectively, combined) complexity of ontology-mediated query answering for a DL $\mathcal{L}$. $\underline{\mathsf{B}}\mathsf{QE}(\mathsf{UCQ}, \mathcal{L})$ is $\mathfrak{C}$-hard and in $\mathrm{PP}^{\mathfrak{C}}$ under the same complexity assumptions.*

Proof. Hardness follows from the fact that BOLs are generalizations of ontology languages: more precisely, if we assume that all axioms are annotated with the global context, we obtain a classical ontology. As for membership, $\underline{\mathsf{B}}\mathsf{QE}(\mathsf{UCQ}, \mathcal{L})$ is in $\mathrm{PP}^{\mathfrak{C}}$ due to the following reason. Given a probabilistic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ a UCQ Q, and a threshold value $p \in [0, 1)$, we construct a nondeterministic Turing machine by creating multiple copies of the worlds $\mathcal{K}_{\mathcal{W}}$ (where $\mathcal{W}$ is a valuation over the variables of the BN) such that they correspond to the nondeterministic branches of a Turing machine and to the uniform distribution over all thus generated worlds. As before, we introduce artificial success (resp., failure) branches into the nondeterministic Turing machine such that satisfying the original threshold corresponds to having a majority of successful computations. Since every branch of the TM corresponds to a KB $\mathcal{K}_{\mathcal{W}} = (\mathcal{T}_{\mathcal{W}}, \mathcal{A}_{\mathcal{W}})$, and OMQA in $\mathcal{L}$ is in $\mathfrak{C}$, every such computation path can decide whether $\mathcal{A}_{\mathcal{W}} \models (\mathcal{T}, Q)$ using a $\mathfrak{C}$ oracle. Then, $\underline{\mathrm{P}}(\mathcal{T}, Q) > p$ is *true* if and only if *more than half* of the computations paths of the nondeterministic TM are accepting. □

We deepen our analysis with the data complexity results. On a rather unsurprising result, we show that Bayesian query answering is as hard as probabilistic inference in BNs, for logics where OMQA is polynomial time in data complexity.

Theorem 8.12 $\underline{\mathsf{B}}\mathsf{QE}(\mathsf{IQ}, \mathcal{L})$ *is* PP*-hard in data complexity.*

Proof. PP-hardness is an immediate consequence of a reduction from probabilistic inference in BNs. Given a BN $\mathcal{B}$, an arbitrary event x, and a threshold value $p \in [0, 1)$, we can reduce the problem of deciding $\mathrm{P}_{\mathcal{B}}(x) > p$ to lower Bayesian query evaluation, as follows. We define the ABox $\mathcal{A}$ which consists of a single probabilistic assertion $\langle A(a) : x \rangle$ and the probabilistic KB $\mathcal{K} = (\emptyset, \mathcal{A}, \mathcal{B})$. Then, we obtain $\underline{\mathrm{P}}(A(a)) > p$ if and only if $\mathrm{P}_{\mathcal{B}}(x) > p$. Note that $A(a)$ is an instance query, which implies that $\underline{\mathsf{B}}\mathsf{QE}(\mathsf{IQ}, \mathcal{L})$ is PP-hard. □

For our data complexity analysis, logics for which OMQA is coNP-complete form the most interesting case in the technical sense. The next result shows that the complexity of probabilistic query evaluation goes one level higher in the *counting polynomial-time hierarchy* for such logics in data complexity.

Theorem 8.13 $\underline{\mathsf{B}}\mathsf{QE}(\mathsf{IQ}, \mathcal{ALC})$ *is* $\mathrm{PP}^{\mathrm{NP}}$*-hard in data complexity.*

Proof. To show hardness, we again reduce from the validity of formulas of the form

$$\Phi \coloneqq \mathsf{C}^c\, x_1, \ldots, x_m \exists y_1, \ldots, y_n\ \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k\,,$$

where every φ_i is a propositional clause over $x_1, \ldots, x_m$, $y_1, \ldots, y_n$, and $k, m, n \geq 1$. As before, we also assume, without loss of generality, that φ contains all clauses of the form $x_j \vee \neg x_j$, $1 \leq j \leq m$, and similarly $y_j \vee \neg y_j$, $1 \leq j \leq n$ and that each clause φ_j contains exactly three literals. We first define the BN $\mathcal{B}_\Phi$ over the variables $x_1 \ldots x_m$ such that $\mathrm{P}_\mathcal{B}(\mathcal{W}) = 0.5^m$. Then, we define the ABox $\mathcal{A}_\Phi$ as follows.

- For each variable x_j, $1 \leq j \leq m$, it contains the assertions $\langle \mathsf{T}(x_j) : x_j \rangle$ and $\langle \mathsf{F}(x_j) : \neg x_j \rangle$, representing the truth assignments of x-variables.
- For each variable y_j, $1 \leq j \leq n$, it contains the assertions $\mathsf{L}(y_j)$.
- Each clause φ_j is described with the help of the binary predicates $\mathsf{pos}_1 \ldots \mathsf{pos}_3$ and $\mathsf{neg}_1 \ldots \mathsf{neg}_3$. For example, consider the clause $\varphi_j = x_2 \vee \neg y_4 \vee y_1$. For the satisfying assignment $x_2 \mapsto \mathit{true}$, $y_4 \mapsto \mathit{true}$, $y_1 \mapsto \mathit{false}$, we add the assertion $\mathsf{pos}_1(j, x_2)$, $\mathsf{pos}_2(j, y_4)$, $\mathsf{neg}_3(j, y_1)$, and similarly for all other satisfying assignments. There are at most 7 satisfying assignments for each clause.

We now define the TBox $\mathcal{T}_\Phi$ containing the axiom

$$\mathsf{L} \sqsubseteq (\mathsf{T} \sqcup \mathsf{F}) \sqcap \neg(\mathsf{T} \sqcap \mathsf{F}),$$

which intuitively assigns each variable y_j, $1 \leq j \leq n$ exactly one truth assignment. To encode the satisfaction condition of the clauses in Φ, we additionally need the axioms

$$\begin{aligned}
&\exists \mathsf{neg}_1.\mathsf{T} \sqcap \exists \mathsf{neg}_2.\mathsf{T} \sqcap \exists \mathsf{neg}_3.\mathsf{T} \sqsubseteq \bot,\\
&\exists \mathsf{neg}_1.\mathsf{T} \sqcap \exists \mathsf{neg}_2.\mathsf{T} \sqcap \exists \mathsf{pos}_3.\mathsf{F} \sqsubseteq \bot,\\
&\qquad \ldots\\
&\exists \mathsf{pos}_1.\mathsf{F} \sqcap \exists \mathsf{pos}_2.\mathsf{F} \sqcap \exists \mathsf{pos}_3.\mathsf{F} \sqsubseteq \bot.
\end{aligned}$$

Then, for the query $\top$, we obtain that Φ is valid if and only if $\mathrm{P}_{\mathcal{K}_\Phi}(\top) > \mathsf{c} \cdot (0.5)^m$, which concludes to the proof. □

An analogous result to Theorem 8.12 can be obtained for the combined complexity: For logics where OMQA is polynomial time in combined complexity, Bayesian query evaluation becomes $\mathrm{PP}^{\mathrm{NP}}$-hard.

Theorem 8.14 *$\underline{\mathsf{B}}\mathsf{QE}(\mathsf{UCQ}, \mathit{DL\text{-}Lite})$ and $\underline{\mathsf{B}}\mathsf{QE}(\mathsf{UCQ}, \mathcal{EL})$ are $\mathrm{PP}^{\mathrm{NP}}$-hard in combined complexity.*

Proof. We again reduce from the validity of formulas of the form

$$\Phi \coloneqq \mathsf{C}^c\, x_1, \ldots, x_m \exists y_1, \ldots, y_n\ \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k\,,$$

where every φ_i is a propositional clause over $x_1, \ldots, x_m$, $y_1, \ldots, y_n$, and $k, m, n \geq 1$. We define the ABox $\mathcal{A}_\Phi$ as follows. For every clause φ_i, and for every satisfying assignment ν to the clause φ_i, we define three probabilistic role assertions, r_i, s_i, and t_i, which

together represent a satisfying assignment. There are 7 satisfying assignments for each clause. For instance, for a clause $\varphi_i = x \vee \neg y \vee z$, we define the assertions

$$\left\langle \mathsf{r}_i(\nu(x), j) : \nu_{|\{x_1,\ldots,x_m\}\cap\{x\}} \right\rangle,$$
$$\left\langle \mathsf{s}_i(\nu(y), j) : \nu_{|\{x_1,\ldots,x_m\}\cap\{y\}} \right\rangle,$$
$$\left\langle \mathsf{t}_i(\nu(z), j) : \nu_{|\{x_1,\ldots,x_m\}\cap\{z\}} \right\rangle,$$

where ν is the j-th truth assignment to the variables x, y, z that satisfies φ_i. and $\nu_{|x_1\ldots x_m}$ denotes the partial assignment which is the restriction of ν to the variables $\{x_1, \ldots, x_m\}$.

We construct the probabilistic KB $\mathcal{K}_\Phi = (\emptyset, \mathcal{A}_\Phi, \mathcal{B}_\Phi)$, where the BN $\mathcal{B}_\Phi$ is defined over the variables $\{x_1, \ldots, x_m\}$ with $\mathrm{P}_\mathcal{B}(\mathcal{W}) = 0.5^m$ for every valuation $\mathcal{W}$ of V and where

$$\mathcal{A}_\Phi = \Big\{ \left\langle \mathsf{r}_i(\nu(x), j) : \nu_{|\{x_1,\ldots,x_m\}\cap\{x\}} \right\rangle, \left\langle \mathsf{s}_i(\nu(y), j) : \nu_{|\{x_1,\ldots,x_m\}\cap\{y\}} \right\rangle, \left\langle \mathsf{t}_i(\nu(z), j) : \nu_{|\{x_1,\ldots,x_m\}\cap\{z\}} \right\rangle \mid \nu \text{ is the } j\text{-th satisfying assignment of } \varphi_i \Big\}.$$

Consider now the following UCQ

$$Q_\Phi = \exists j_1 \ldots j_k, x_1, \ldots x_m, y_1, \ldots y_n, \bigwedge_{i=1}^{k} \mathsf{r}_i(x, j_i) \wedge \mathsf{s}_i(y, j_i) \wedge \mathsf{t}_i(z, j_i),$$

which lifts the satisfaction condition of Φ: For every satisfying assignment ν to Φ there is a corresponding world that satisfies the query Q_Φ and vice versa. Then, it is easy to verify that

$$\underline{\mathrm{P}}_{\mathcal{K}'}(Q_\Phi) > c \cdot 0.5^m \text{ if and only if } \Phi \text{ is valid.}$$

Observe that the above reduction can clearly be done in polynomial time in the size of Φ, and that the resulting TBox is empty. Moreover, the assertions used in the construction are all positive and simple; thus, the hardness applies to both *DL-Lite* and $\mathcal{EL}$. □

Observe that all membership results follow from Theorem 8.11 except for $\mathcal{SHOIQ}$. Importantly, instance query answering is coNExp-complete for $\mathcal{SHOIQ}$ and we obtain $\mathrm{PP}^{\mathrm{coNExp}}$ upper bound as a result of Theorem 8.11. We note that the non-determinism in the oracle coNExp calls are used in a restricted fashion (as in Theorem 7.9) and obtain that $\underline{\mathsf{B}}\mathsf{QE}(\mathsf{IQ}, \mathcal{SHOIQ})$ is in coNExp in combined complexity.

All of our results for lower Bayesian query evaluation are summarized in Table 8.1. All of our results are in their most general form. i.e., all of the technical reductions yield tight complexity bounds for lower Bayesian query answering for all logics under consideration, with the only exception being $\mathcal{SHOIQ}$. Therefore, the only cases that we leave open are for $\mathcal{SHOIQ}$ is for unions of conjunctive queries.The precise complexity of OMQA in $\mathcal{SHOIQ}$ for unions of conjunctive queries is a long standing open problem.

The data complexity results show an interesting pattern. Interestingly, all hardness results hold already for instance queries. Arguably, the most notable result is the $\mathrm{PP}^{\mathrm{NP}}$-hardness of $\underline{\mathsf{B}}\mathsf{QE}(\mathsf{IQ}, \mathcal{ALC})$, which clearly transfers to $\underline{\mathsf{B}}\mathsf{QE}(\mathsf{UCQ}, \mathcal{ALC})$. Putting aside the case of $\mathcal{SHOIQ}$, all membership results are the same for instance queries and

Table 8.1: Complexity results for *lower* BQE relative to different DLs.

Description Logic	Query Evaluation in BOLs			
	Instance Queries		Unions of Conjunctive Queries	
	data	**combined**	**data**	**combined**
DL-Lite, *DL-Lite*$_\mathcal{R}$	PP	PP	PP	PP^{NP}
$\mathcal{EL}$, $\mathcal{ELH}$	PP	PP	PP	PP^{NP}
$\mathcal{ELI}$, Horn-$\mathcal{SHOIQ}$	PP	EXP	PP	EXP
$\mathcal{ALC}$, $\mathcal{ALCHQ}$	PP^{NP}	EXP	PP^{NP}	EXP
$\mathcal{ALCI}$, $\mathcal{SH}$, $\mathcal{SHIQ}$	PP^{NP}	EXP	PP^{NP}	2EXP
$\mathcal{SHOIQ}$	PP^{NP}	coNEXP	$\geq \text{PP}^{\text{NP}}$	$\geq$ coN2EXP

unions of conjunctive queries. The combined complexity results, on the other side, differ significantly depending on whether we consider instance queries or conjunctive queries. For example, while $\underline{\text{B}}\text{QE}(\text{IQ}, \mathcal{EL})$ is PP-complete (by Theorem 8.12), $\underline{\text{B}}\text{QE}(\text{UCQ}, \mathcal{EL})$ is PP^{NP}-complete and a similar phenomenon is observed also for *DL-Lite*, *DL-Lite*$_\mathcal{R}$ and $\mathcal{ELH}$.

There are more insights regarding the complexity of Bayesian query evaluation. Note that a major advantage of BOLs is the re-usability of the context variables for different axioms. Intuitively, this means that a large portion of the ontology may be dependent on similar factors (specified by the same context variables) and as a consequence the size of the BN does not necessarily grow proportionally with the size of the data. This makes a significant difference also in the theoretical sense since $\text{PP}^{\mathfrak{C}}$-hardness is in the size of the network. Therefore, if an application can ensure that the size of the BN does not grow with the data, or is fixed, then the complexity of Bayesian query evaluation will be the same as OMQA. In particular, for DLs such as $\mathcal{EL}$, this implies fixed-parameter tractability in data complexity. These observations have led to a proof-of-concept reasoner (BORN) for Bayesian ontology languages (Ceylan, Mendez, and Peñaloza 2015), which needs to be optimized further.

Complexity Results for $\overline{\text{B}}\text{QE}$

We now discuss how our complexity results can be extended to upper Bayesian query evaluation. Our first result illustrates a connection between the decision problems $\underline{\text{P}}(Q) > p$ and $\overline{\text{P}}(Q) > p$. More precisely, we show that checking $\overline{\text{P}}(Q) > p$ is at most as hard as checking $\underline{\text{P}}(\bot) \leq p$.

Theorem 8.15 *Let $p \in (0, 1]$ be a threshold value, $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ be a probabilistic knowledge base, and Q be a Boolean query. Deciding whether $\overline{\text{P}}_\mathcal{K}(Q) > p$ can always be reduced in polynomial time to deciding whether $\underline{\text{P}}_{\mathcal{K}'}(\bot) \leq p'$ for some probabilistic knowledge base $\mathcal{K}'$ and threshold p'.*

Proof. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ be a probabilistic KB, $Q = Q_1 \vee \ldots \vee Q_n$ a query and $\mathsf{gr}(Q_i)$ be the set of atoms from a grounding of Q_i with fresh constants. By Theorem 8.9, we know that $\overline{\text{P}}_\mathcal{K}(Q) = 1 - \underline{\text{P}}_\mathcal{K}(\neg Q)$. We now describe a procedure for deciding whether $\underline{\text{P}}_\mathcal{K}(\neg Q) \leq p$, which in turn can be used to for deciding $\overline{\text{P}}_\mathcal{K}(Q) > p$.

Table 8.2: Complexity results for *upper* BQE relative to different DLs. Notice that the upper probability is always 1 for the BOLs based on languages $\mathcal{EL}$, $\mathcal{ELH}$ and $\mathcal{ELI}$ since they can not express negative information. $\mathcal{EL}_\bot$, $\mathcal{ELH}_\bot$ and $\mathcal{ELI}_\bot$ denote the DLs that extend these languages with $\bot$.

Description Logic	**Query Evaluation in BOLs**	
	IQs and UCQs	
	data	**combined**
DL-Lite, *DL-Lite*$_\mathcal{R}$	PP	PP
$\mathcal{EL}_\bot$, $\mathcal{ELH}_\bot$	PP	PP
$\mathcal{ELI}_\bot$, Horn-$\mathcal{SHOIQ}$	PP	EXP
$\mathcal{ALC}$, $\mathcal{ALCHQ}$	PP^{NP}	EXP
$\mathcal{ALCI}$, $\mathcal{SH}$, $\mathcal{SHIQ}$	PP^{NP}	EXP
$\mathcal{SHOIQ}$	PP^{NP}	NEXP

Let $\mathcal{K}' = (\mathcal{T}', \mathcal{A}', \mathcal{B}')$ be a probabilistic KB where $\mathcal{A}'$ contains assertions of the form $\langle a : q_i \rangle$ for all atoms $a \in \mathsf{gr}(Q_i)$ and for $1 \leq i \leq n$. Moreover, $\mathcal{A}'$ contains, for all probabilistic assertions $\langle a : \varphi \rangle \in \mathcal{A}$, an assertion of the form $\langle a : \varphi \wedge (q_1 \vee \ldots \vee q_n) \rangle$. Accordingly, the BN $\mathcal{B}'$ extends the variables V of $\mathcal{B}$ to V' which contains the additional variables $q_1, \ldots, q_n$. Intuitively, the BN $\mathcal{B}'$ differs from the BN $\mathcal{B}$ only in the sense that $\mathcal{B}'$ ensures that $\text{P}_{\mathcal{B}'}(q_i \mid q_j) = 0$ where $i \neq j$ and $\text{P}_{\mathcal{B}'}(q_i) = 1/n$ for all newly introduced variables. Notice that the construction of $\mathcal{B}'$ can be exponential in the size of the query if preformed naïvely. However, using auxiliary nodes, this construction can be done in size polynomial of the given BN $\mathcal{B}$, as in (Roth 1996).

Observe that, for all valuations $\mathcal{W}$ of the variables V, the entailment relation $\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, \neg Q)$ in $\mathcal{K}$ holds if and only there exists a valuation $\mathcal{W}' \models \mathcal{W}$ of the variables V' such that $\mathcal{A}'_{\mathcal{W}'} \models (\mathcal{T}_{\mathcal{W}'}, \bot)$ in $\mathcal{K}'$. Using this and the fact that, $\text{P}_{\mathcal{B}'}(\mathcal{W}') = \text{P}_\mathcal{B}(\mathcal{W}) \cdot 1/n$, we obtain

$$\underline{\text{P}}_\mathcal{K}(\neg Q) = \sum_{\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, \neg Q)} \text{P}_\mathcal{B}(\mathcal{W}) = n \cdot \sum_{\mathcal{A}'_{\mathcal{W}'} \models (\mathcal{T}_{\mathcal{W}'}, \bot)} \text{P}_\mathcal{B}(\mathcal{W}'),$$

which implies that we can decide $\underline{\text{P}}_\mathcal{K}(\neg Q) \leq p$ by deciding $\underline{\text{P}}_{\mathcal{K}'}(\bot) \leq n \cdot p$. □

Theorem 8.15 helps us to directly transfer some results from $\underline{\text{B}}$QE to $\overline{\text{B}}$QE. First of all, it is sufficient to analyze the complexity of $\underline{\text{P}}_\mathcal{K}(\bot) \leq p$. Intuitively, this means that, we need to sum over the probabilities of the inconsistent worlds. Notice that this is a trivial problem for logics which can not express negative information. For example, for the DL $\mathcal{EL}$, the upper probability of a query will always be 1, which is an expected behavior

What about the other languages and the extensions of $\mathcal{EL}$, $\mathcal{ELH}$ and $\mathcal{ELI}$ with $\bot$? For all these languages, the complexity of checking inconsistency, in our notation $\mathcal{A} \models (\mathcal{T}, \bot)$, and the complexity of instance query answering is the same. As a result, we obtain a general upper bound of $\text{PP}^\mathfrak{C}$ for $\overline{\text{B}}\mathsf{QE}(\mathsf{UCQ}, \mathcal{L})$, where $\mathfrak{C}$ denotes the complexity of inconsistency checking (same as instance query answering) in $\mathfrak{C}$. This is an immediate consequence of Theorem 8.11 and Theorem 8.15. Note that this implies the same complexity results as for $\underline{\text{B}}$QE for all cases (since the hardness results apply modulo

minor modifications). Notably, the difference between instance queries and unions of conjunctive queries disappears for $\overline{\mathsf{B}}\mathsf{QE}$ as summarized in Table 8.2

Remark. Given the fact that upper Bayesian query evaluation can be seen as a special case of lower Bayesian query evaluation, in the remainder of this chapter, we will only focus on the *lower* Bayesian query evaluation for ease of presentation. Without further notice, we simply write $\mathrm{P}_{\mathcal{K}}(Q)$, or BQE instead of $\underline{\mathrm{P}}_{\mathcal{K}}(Q)$, or $\underline{\mathsf{B}}\mathsf{QE}$ and all the remaining results are based on this assumption.

8.1.2 Incorporating Evidence to Ontological Queries

We have introduced Bayesian query evaluation and studied its computational complexity. So far, we have not provided any means for incorporating *evidence*. We now show that evidence can be incorporated into these problems in terms of conditional probabilities.

Formally, we define an evidence as a conjunction of elementary events over the BN and denote it by $\mathbf{e}$. Note that evidence can also be viewed as a partial valuation and therefore the probability of the evidence itself can be computed as in BNs. Based on these, we now define how to compute the conditional probability of a query given an evidence.

Definition 8.16 (query evaluation with evidence) Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ be a probabilistic KB, Q a Boolean Query and $\mathbf{e}$ an evidence. The probability of a Q given $\mathbf{e}$ w.r.t. a probabilistic interpretation $\mathbf{I} = (\mathfrak{I}, \mathrm{P}_{\mathfrak{I}})$, denoted as $\mathrm{P}_{\mathbf{I}}(Q \mid \mathbf{e})$, is given by

$$\mathrm{P}_{\mathbf{I}}(Q \mid \mathbf{e}) \times \mathrm{P}_{\mathbf{I}}(\mathbf{e}) = \mathrm{P}_{\mathbf{I}}(Q \wedge \mathbf{e})$$

where

$$\mathrm{P}_{\mathbf{I}}(Q \wedge \mathbf{e}) = \sum_{\mathcal{I} \in \mathfrak{I}, \mathcal{I} \models Q, \mathcal{V}^{\mathcal{I}} \models \mathbf{e}} \mathrm{P}_{\mathfrak{I}}(\mathcal{I}) \quad \text{and} \quad \mathrm{P}_{\mathbf{I}}(\mathbf{e}) = \mathrm{P}_{\mathcal{B}}(\mathbf{e}).$$

The *probability* of a Boolean query Q given $\mathbf{e}$, denoted $\mathrm{P}_{\mathcal{K}}(Q \mid \mathbf{e})$ is given by

$$\mathrm{P}_{\mathcal{K}}(Q \mid \mathbf{e}) := \inf_{\mathbf{I} \models \mathcal{K}} \mathrm{P}_{\mathbf{I}}(Q \mid \mathbf{e}).$$

The corresponding decision problem is defined as before. ◊

Being able to incorporate evidence is a fundamental task in probabilistic models as such evidence may significantly change the query probability we are interested in. Consider, for instance, our running example and suppose that we observed by some sensor readings that bob has Hypertension. This observation clearly increases the probability of bob being in a CriticalSituation. In fact, it holds that

$$\mathrm{P}_{\mathcal{K}_h}(Q_{\mathsf{bob}} \mid \mathsf{s}) = .361 > \mathrm{P}_{\mathcal{K}_h}(Q_{\mathsf{bob}}) = .294$$

Clearly, Bayesian query evaluation under evidence is at least as hard as the case without evidence since the latter is a special case of the former. Moreover, from Theorem 8.8, it is easy to see that

$$\mathrm{P}_{\mathcal{K}}(Q \mid \mathbf{e}) \times \mathrm{P}_{\mathcal{B}}(\mathbf{e}) = \sum_{\substack{\mathcal{A}_{\mathcal{W}} \models (\mathcal{T}_{\mathcal{W}}, Q) \\ \mathcal{W} \models \mathbf{e}}} \mathrm{P}_{\mathcal{B}}(\mathcal{W})$$

for a probabilistic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$. Intuitively, we can employ all techniques developed for query evalution in BOLs to compute $\mathrm{P}_{\mathcal{K}}(Q \mid \mathbf{e})$ with the difference that we have to consider the intersection of the worlds to incorporate evidence.

Given these and the fact that PP (and also $\mathrm{PP}^{\mathrm{NP}}$, Exp, 2Exp, and coNExp) is closed under intersection and union (Beigel, Reingold, and Spielman 1995), we can generalize all our results for Bayesian query evaluation under evidence.

Corollary 8.17 *The complexity of Bayesian query evaluation under evidence is the same as the complexity of Bayesian query evaluation for all languages $\mathcal{L}$ given in Table 8.1.*

8.1.3 Most Likely Contexts for Ontological Queries

As pointed earlier, Bayesian ontology languages can be viewed as a generalization of BNs. As a consequence, standard inference problems in BNs transfer to BOLs in a natural way. So far, we only focused on the query evaluation problem, which intuitively extends probabilistic inference in Bayesian networks by incorporating first-order knowledge.

In this section, we study the problem of identifying the *most likely context* in Bayesian ontology languages. Informally, finding the most likely context for a query can be seen as the dual of computing the probability of a query. Intuitively, we are interested in finding the context with the highest probability in which the query is entailed. As a special case, we are interested in identifying the *most likely world* for a query.

Obviously, these problem are lifted from MAP and MPE in BNs (Koller and Friedman 2009; Park and Darwiche 2004b; Pearl 1988). We note that these problems are also related to MPD and MPH (see Chapter 5) in a conceptual sense, but are different in the technical sense. In Bayesian ontology languages, explanations are in terms of propositional formulas (instead of database atoms), which are encoded in a BN (which, unlike a tuple-independent PDB, encodes conditional dependencies).

Moreover, although a context can be in any form; for the problem of most likely context, we restrict our attention to *conjunctive formulas*, which are simply a conjunction of (positive or negative) propositional literals. For a probabilistic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$, a query Q and a context φ, we define

$$\mathcal{A}_\varphi \models (\mathcal{T}_\varphi, Q) \text{ if and only if } \mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q) \text{ for all } \mathcal{W} \models \varphi,$$

which formalized the condition whether a query is entailed from a context φ. Given these preliminaries, most likely context is defined as follows.

Definition 8.18 (most likely context) Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ be a probabilistic KB and Q a UCQ. The *most likely context* for a UCQ Q over a KB $\mathcal{K}$ is given by

$$\arg\max_{\varphi} \Big\{ \sum_{\substack{\mathcal{W} \models \varphi \\ \mathcal{A}_\mathcal{W} \not\models (\mathcal{T}_\mathcal{W}, \bot)}} \mathrm{P}_\mathcal{B}(\mathcal{W}) \;\mid\; \mathcal{A}_\varphi \models (\mathcal{T}_\varphi, Q) \Big\}, \tag{8.6}$$

where φ is a *conjunctive formula* over the variables of the BN. If the context φ is a world, i.e., it contains a literal for all variables in the BN, then this problem is called the *most likely world* for a UCQ Q over a KB $\mathcal{K}$ and simplifies to

$$\arg\max_{\mathcal{W}} \{ \mathrm{P}_\mathcal{B}(\mathcal{W}) \mid \mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q) \text{ and } \mathcal{K}_\mathcal{W} \text{ is consistent.} \}. \tag{8.7}$$

$\Diamond$

Note that the definition for most likely context is slightly different from previous work (Ceylan and Peñaloza 2014b) in the sense that we now require only the consistent worlds to entail the query. Note also that, for logics, which cannot express inconsistency (such as $\mathcal{EL}$), this definition naturally simplifies. The associated decision problems are then defined as follows.

Definition 8.19 (MLC and MLW) Given a probabilistic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$, a UCQ Q, and a threshold value p, MLC (resp., MLW) is to determine whether there exists a context φ (resp., world $\mathcal{W}$) that satisfies Condition (8.6) (resp. Condition (8.7)) with a threshold at least p. MLC (resp., MLW) is parametrized with the language of the ontology and the query; we write $\mathsf{MLC}(\mathsf{Q}, \mathcal{L})$ (resp., $\mathsf{MLW}(\mathsf{Q}, \mathcal{L})$) to refer to the problem on the class Q of queries and on the class of ontologies restricted to the language $\mathcal{L}$. ◊

Both of these reasoning tasks can be potentially useful for diagnosing the observations given in terms of queries. We briefly illustrate the introduced notions on our running example.

Example 8.20 Consider again the KB $\mathcal{K}_h = (\mathcal{T}_h, \mathcal{A}_h, \mathcal{B}_h)$. Suppose that we have observed that bob is in critical situation, or equivalently the query

$$Q_{\mathsf{bob}} = \exists y\ \mathsf{Patient}(\mathsf{bob}), \mathsf{risk}(\mathsf{bob}, y), \mathsf{CriticalSituation}(y).$$

and we are interested in diagnosing this observation, finding the most likely context, which satisfies this query. It is easy to see that the context $\mathsf{s} \wedge \mathsf{c}$ is the most likely context for Q in $\mathcal{K}$. More precisely, $\mathcal{A}_{\mathsf{s}\wedge\mathsf{c}} \models (\mathcal{T}_{\mathsf{s}\wedge\mathsf{c}}, Q)$, all (consistent) worlds that extend φ entail Q, and their probabilities add up to .294. Note that neither the context s nor c provide enough information to entail Q_{bob}. ◊

Observe also that the most likely context is a *partial* explanation, i.e., the conjunctive formula can be a partial valuation over the variables of the BN. The problem of finding the most likely world is a special case where explanations must be complete valuations. Although partial explanations are more informative; in general, they are more intractable compared to complete explanations. We analyze the computational complexity of these problems in detail.

Complexity Results for Most Likely World

We start our technical analysis with the problem of most likely world. In a broad sense, to be able to produce the correct answer to the problem of most likely world, we need to identify a valuation $\mathcal{W}$ such that the conditions

$$i.\ \mathrm{P}_{\mathcal{B}}(\mathcal{W}) > p, \qquad ii.\ \mathcal{A}_{\mathcal{W}} \not\models (\mathcal{T}_{\mathcal{W}}, \bot), \qquad iii.\ \mathcal{A}_{\mathcal{W}} \models (\mathcal{T}_{\mathcal{W}}, Q)$$

are satisfied. The first test can be done in polynomial time, which clearly separates this problem from most likely context in terms of computational complexity. The second test is a *consistency check* and the third one is OMQA and the complexity of these tests depend on the underlying ontology language.

Although seemingly similar, it is worth to note that *consistency checking* and OMQA do not necessarily have the same computational complexity. In fact, these tasks typically

belong to the dual complexity classes. For instance, OMQA in $\mathcal{ALC}$ is coNP-complete (even for instance queries) in data complexity whereas consistency checking in $\mathcal{ALC}$ is NP-complete in data complexity. These observations will play a critical role in our results.

Observe, first, that we can design a naïve algorithm, which guesses a world $\mathcal{W}$ and performs the tests *i–iii* using an appropriate oracle.

Theorem 8.21 *Let $\mathfrak{C}$ be the data (resp., combined) complexity of OMQA in the DL $\mathcal{L}$. Then,* $\mathsf{MLW}(\mathsf{IQ}, \mathcal{L})$ *is $\mathfrak{C}$-hard and* $\mathsf{MLW}(\mathsf{UCQ}, \mathcal{L})$ *in* $\mathrm{NP}^{\mathfrak{C}}$ *in data (resp., combined) complexity.*

Proof. Hardness is immediate since BOLs are proper generalizations of the underlying ontology language. As for membership, given a probabilistic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ a UCQ Q, and a threshold value $p \in [0, 1)$, we consider a nondeterministic Turing machine with a $\mathfrak{C}$ oracle. We guess a world $\mathcal{W}$ and verify with the oracle $\mathfrak{C}$ that i) $\mathrm{P}_{\mathcal{B}}(\mathcal{W}) > p$, ii) $\mathcal{A}_{\mathcal{W}} \not\models (\mathcal{T}_{\mathcal{W}}, \bot)$, and iii) $\mathcal{A}_{\mathcal{W}} \models (\mathcal{T}_{\mathcal{W}}, Q)$. Note that the first test can be done in polynomial time and the third test is in $\mathfrak{C}$ by assumption. The second test, i.e., checking the consistency of the knowledge base, is *at most* as hard as the *complement* of $\mathfrak{C}$ in all languages given in Table 6.4, which implies that this tests can also be performed in the oracle, which proves membership. □

Observe that for logics which admit a polynomial time algorithm for OMQA (and thus for consistency checking) in the data (resp., combined) complexity, this result implies an NP upper bound for MLW in data (resp., combined) complexity. We now show a lower bound.

Theorem 8.22 $\mathsf{MLW}(\mathsf{IQ}, \mathcal{L})$ *is* NP*-hard in data complexity.*

Proof. We show NP-hardness by a reduction from MPE in BNs (see Definition 5.4). Let $\mathcal{B} = (G, \Theta)$ be the given BN, and μ a partial valuation of the variables V, and p a threshold value. For the reduction, we define the probabilistic KB $\mathcal{K} = (\emptyset, \mathcal{A}, \mathcal{B})$ where $\mathcal{B}$ is identical to the given BN and

$$\mathcal{A} = \{\langle \mathsf{A}(v) : v \wedge \mu \rangle, \langle \mathsf{A}(v) : \neg v \wedge \mu \rangle \mid \text{ for all } v \in V \text{ that do not appear in } \mu\}.$$

Furthermore, for the query $Q \coloneqq \top$, there exists a most likely world for $\mathcal{W}$ in $\mathcal{K}$ with threshold more than p if and only if there exists a valuation $\mathcal{W}$ of the variables in V that extends μ such that $\mathrm{P}_{\mathcal{B}}(\mathcal{W}) > p$. Observe that the construction is polynomial and it only uses assertions; thus, yields NP-hardness for all languages, we consider. □

This result is rather unsurprising given the NP-completeness of MPE in BNs (Shimony 1994). In fact, the reduction applies to atomic queries and does not use any TBox axioms. For languages where OMQA is coNP-complete in data complexity, we obtain a Σ_2^{P} upper bound as a consequence of Theorem 8.21. We show that this is indeed a matching lower bound.

Theorem 8.23 $\mathsf{MLW}(\mathsf{IQ}, \mathcal{ALC})$ *is Σ_2^{P}-hard in data complexity.*

Proof. We reduce from the validity of formulas of the form

$$\Phi = \exists x_1, \ldots, x_n \, \forall y_1, \ldots, y_m \, \varphi,$$

where $\varphi = \varphi_1 \vee \cdots \vee \varphi_k$ is a propositional formula in 3DNF. As before, we assume, without loss of generality, that φ contains all clauses of the form $x_j \wedge \neg x_j$, $1 \leq j \leq n$, and similarly $y_j \wedge \neg y_j$, $1 \leq j \leq m$ and that each clause φ_j contains exactly three literals. We first define the BN $\mathcal{B}_\Phi$ over the variables $x_1 \dots x_n$ such that $\mathrm{P}_\mathcal{B}(\mathcal{W}) = 0.5^n$. Then, we define the ABox $\mathcal{A}_\Phi$ as follows.

- It contains the the assertions $\mathsf{J}(1), \dots, \mathsf{J}(k)$. We later use these assertions to ensure that all clauses φ_j, $1 \leq j \leq k$ are visited.
- For each variable x_j, $1 \leq j \leq n$, it contains the assertions $\langle \mathsf{T}(x_j) : x_j \rangle$ and $\langle \mathsf{F}(x_j) : \neg x_j \rangle$, representing the truth assignments of x-variables.
- For each variable y_j, $1 \leq j \leq m$, it contains the assertions $\mathsf{L}(y_j)$.
- Each clause φ_j is described with the help of the binary predicates $\mathsf{pos}_1, \dots, \mathsf{pos}_3$ and $\mathsf{neg}_1, \dots, \mathsf{neg}_3$. For example, consider the clause $\varphi_j = x_2 \wedge \neg y_4 \wedge y_1$. For the satisfying assignment $x_2 \mapsto \mathit{true}$, $y_4 \mapsto \mathit{false}$, $y_1 \mapsto \mathit{true}$, we add the assertions $\mathsf{pos}_1(j, x_2)$, $\mathsf{neg}_2(j, y_4)$, $\mathsf{pos}_3(j, y_1)$.

We now define the TBox $\mathcal{T}_\Phi$ containing the axiom

$$\mathsf{L} \sqsubseteq (\mathsf{T} \sqcup \mathsf{F}) \sqcap \neg(\mathsf{T} \sqcap \mathsf{F}),$$

which intuitively assigns each variable y_j, $1 \leq j \leq n$ exactly one truth assignment. To encode the satisfaction condition of the clauses in Φ, we additionally need the axioms

$$\begin{aligned}
&\exists \mathsf{neg}_1.\mathsf{F} \sqcap \exists \mathsf{neg}_2.\mathsf{F} \sqcap \exists \mathsf{neg}_3.\mathsf{F} \sqsubseteq \mathsf{A},\\
&\exists \mathsf{neg}_1.\mathsf{F} \sqcap \exists \mathsf{neg}_2.\mathsf{F} \sqcap \exists \mathsf{pos}_3.\mathsf{T} \sqsubseteq \mathsf{A},\\
&\qquad \dots\\
&\exists \mathsf{pos}_1.\mathsf{T} \sqcap \exists \mathsf{pos}_2.\mathsf{T} \sqcap \exists \mathsf{pos}_3.\mathsf{T} \sqsubseteq \mathsf{A}.
\end{aligned}$$

Finally, we make sure that every clause is satisfied by the axiom

$$\neg \mathsf{A} \sqcap \mathsf{J} \sqsubseteq \bot.$$

Then, for the query $\top$, we obtain that Φ is valid if and only if there exists a world $\mathcal{W}$ such that $\mathcal{A}_\Phi \models (\mathcal{T}_\Phi, \top)$ and $\mathrm{P}(\mathcal{W}) > (0.5)^n$. □

It only remains to observe that Theorem 8.21 yields tight complexity bounds for all remaining logics except for $\mathcal{SHOIQ}$. Observe also that $\mathsf{MLW}(\mathsf{IQ}, \mathcal{SHOIQ})$, is in $\mathrm{NP}^{\mathrm{NEXP}}$ by Theorem 8.21, which equals to P^{NE}. Altogether, these results are summarized in Table 8.3.

Complexity Results for Most Likely Context

Most likely context generalizes most likely world by allowing partial explanations. As before, we can employ a guess-check algorithm: we can guess a context φ and perform the entailment and consistency tests using an appropriate oracle. However, as contexts are partial, we now have to consider all extensions $\mathcal{W}$ of φ, where $\mathcal{K}_\mathcal{W}$ is consistent, and add up the probability of this world if it entails the query. This suggests the need for a stronger oracle.

Table 8.3: Complexity results for most likely world.

Description Logics	Most Likely World			
	Instance Queries		Unions of Conjunctive Queries	
	data	**combined**	**data**	**combined**
DL-Lite, *DL-Lite*$_\mathcal{R}$	NP	NP	NP	NP
$\mathcal{EL}$, $\mathcal{ELH}$	NP	NP	NP	NP
$\mathcal{ELI}$, Horn-$\mathcal{SHOIQ}$	NP	EXP	NP	EXP
$\mathcal{ALC}$, $\mathcal{ALCHQ}$	Σ_2^{P}	EXP	Σ_2^{P}	EXP
$\mathcal{ALCI}$, $\mathcal{SH}$, $\mathcal{SHIQ}$	Σ_2^{P}	EXP	Σ_2^{P}	2EXP
$\mathcal{SHOIQ}$	Σ_2^{P}	$\leq \mathrm{P}^{\mathrm{NE}}$	$\geq \Sigma_2^{\mathrm{P}}$	$\geq$ CON2EXP

Theorem 8.24 *Let $\mathfrak{C}$ denote the data (respectively, combined) complexity of OMQA for a DL language $\mathcal{L}$. $\mathsf{MLC}(\mathsf{UCQ}, \mathcal{L})$ is $\mathfrak{C}$-hard and in $\mathrm{NP}^{\mathrm{PP}^{\mathfrak{C}}}$ in the data (respectively, combined) complexity.*

Proof. Hardness is immediate as in the proof of Theorem 8.21. For membership, given a probabilistic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ a UCQ Q, and a threshold value $p \in [0, 1)$, we consider a nondeterministic Turing machine with a $\mathrm{PP}^{\mathfrak{C}}$ oracle. We guess a context φ and verify with the oracle $\mathrm{PP}^{\mathfrak{C}}$ that

i. $\sum_{\mathcal{W} \models \varphi, \mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q)} \mathrm{P}_\mathcal{B}(\mathcal{W}) > p$,
ii. $\mathcal{A}_\mathcal{W} \not\models (\mathcal{T}_\mathcal{W}, \bot)$ for any world $\mathcal{W} \models \varphi$,
iii. $\mathcal{A}_\mathcal{W} \models (\mathcal{T}_\mathcal{W}, Q)$ for all worlds $\mathcal{W} \models \varphi$.

Note that the first test can be done in PP. The second and the third test is in $(\mathrm{CO})\mathrm{NP}^{\mathfrak{C}}$ and thus in $\mathrm{PP}^{\mathfrak{C}}$, which implies that these tests can also be performed in the oracle, which concludes the result. □

Let us briefly discuss the implications of Theorem 8.24. First of all, we note that for all cases where $\mathfrak{C} = \mathrm{EXP}$, or $\mathfrak{C} = 2\mathrm{EXP}$, we immediately obtain tight complexity bounds for MLC. In all remaining cases (excluding $\mathcal{SHOIQ}$), we obtain an $\mathrm{NP}^{\mathrm{PP}}$ upper bound. In the following, we show a matching lower bound for all these languages.

Theorem 8.25 *$\mathsf{MLC}(\mathsf{IQ}, \mathcal{L})$ is $\mathrm{NP}^{\mathrm{PP}}$-hard in data complexity.*

Proof. We show $\mathrm{NP}^{\mathrm{PP}}$-hardness by a reduction from MAP in BNs, that is, given a BN $\mathcal{B} = (G, \Theta)$, a partial valuation μ of the variables V, a set $X \subseteq V$, and a threshold value $p \in [0, 1)$, is there a valuation $\mathcal{W}$ of the variables in X that extends μ such that $\mathrm{P}(\mathcal{W} \wedge \mu) > p$? Let the set $\{x_1 \dots x_n\}$ denote the variables in X which do not appear in μ. For the reduction, we define the probabilistic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ where $\mathcal{B}$ is identical to the given BN. Moreover, the TBox $\mathcal{T}$ consists of only the axiom $\exists \mathsf{r}.\top \sqsubseteq \mathsf{A}$, and the ABox $\mathcal{A}$ contains the role assertions

$$\mathsf{r}(1, 2),\ \mathsf{r}(2, 3), \dots \mathsf{r}(n-1, n),$$

and finally for every x_i, $1 \leq i \leq n$, the concept assertions $\langle \mathsf{A}(i) : x_i \wedge \mu \rangle, \langle \mathsf{A}(i) : \neg x_i \wedge \mu \rangle$. Then, the atomic query $Q \coloneqq \mathsf{A}(n)$ is entailed only from those worlds where the assertions

Table 8.4: Complexity results for most likely context.

Description Logics	Most Likely Context			
	Instance Queries		Unions of Conjunctive Queries	
	data	**combined**	**data**	**combined**
DL-Lite, *DL-Lite*$_\mathcal{R}$	NP$^{\text{PP}}$	NP$^{\text{PP}}$	NP$^{\text{PP}}$	NP$^{\text{PP}}$
$\mathcal{EL}$, $\mathcal{ELH}$	NP$^{\text{PP}}$	NP$^{\text{PP}}$	NP$^{\text{PP}}$	NP$^{\text{PP}}$
$\mathcal{ELI}$, Horn-$\mathcal{SHOIQ}$	NP$^{\text{PP}}$	Exp	NP$^{\text{PP}}$	Exp
$\mathcal{ALC}$, $\mathcal{ALCHQ}$	NP$^{\text{PP}}$	Exp	NP$^{\text{PP}}$	Exp
$\mathcal{ALCI}$, $\mathcal{SH}$, $\mathcal{SHIQ}$	NP$^{\text{PP}}$	Exp	NP$^{\text{PP}}$	2Exp
$\mathcal{SHOIQ}$	NP$^{\text{PP}}$	$\leq$ P$^{\text{NE}}$	$\geq$ NP$^{\text{PP}}$	$\geq$ coN2Exp

$\mathsf{A}(1)$, $\mathsf{A}(1)$, …, $\mathsf{A}(n-1)$ hold since the TBox axiom forces this through a sequence of role assertions encoded with r. Furthermore, by construction, each such assertion can only be satisfied in a context, which extends μ, and sets, each x_i either to true or false. Thus, we conclude that there exits a most likely context for Q in $\mathcal{K}$ with threshold exceeding p if and only if there a valuation $\mathcal{W}$ of the variables in X that extends μ such that $\mathrm{P}_\mathcal{B}(\mathcal{W} \wedge \mu) > p$. □

$\mathsf{MLC}(\mathsf{IQ}, \mathcal{SHOIQ})$ is in NP$^{\text{NExp}}$ by Theorem 8.24, which equals to P$^{\text{NE}}$. All of our results are completeness results with the only exception being $\mathcal{SHOIQ}$. These results are summarized in Table 8.4.

8.2 Dynamic Bayesian Ontology Languages

Verifying dynamic systems has been a long endeavor in DLs, which has led to the study of temporal DLs (Artale and Franconi 2005; Baader, Ghilardi, and Lutz 2012; Lutz, Wolter, and Zakharyaschev 2008). For a review on *ontology-based monitoring*, we refer the reader to (Baader 2014). Lately, these research efforts have been shifted towards more data-oriented systems, which are more in line with the spirit of ontology-mediated query answering (Artale, Kontchakov, Kovtunova, Ryzhikov, Wolter, and Zakharyaschev 2015; Artale, Kontchakov, Ryzhikov, and Zakharyaschev 2015; Artale, Kontchakov, Wolter, and Zakharyaschev 2013; Baader, Borgwardt, and Lippmann 2013; Borgwardt, Lippmann, and Thost 2013).

Verification with temporal DLs distinguishes itself from classical model checking by allowing open-world reasoning and advanced querying tools. One major drawback of ontology based monitoring is, however, its limitations for making projections about the future states of a system. Existing proposals verify a dynamic system relative to certain conditions (such as rigidity), and check whether these conditions can be satisfied in one (or all) successful runs. When it comes to making concrete projections about the future states of a system, they are rather unsatisfactory.

Dynamic Bayesian networks (DBNs) (Dean and Kanazawa 1989; Murphy 2002), on the other hand, are structures specifically tailored towards modeling stochastic processes. DBNs generalize hidden Markov models and Kalman filters and as such have been suc-

cessfully employed in many different areas including *robotics*, *speech recognition*, *weather forecasting* (Dagum, Galper, and Horvitz 1992), *bio-medical applications* (Husmeier 2003; Nachimuthu and Haug 2012), *healthcare management*, and *diagnosis* (Choi, Darwiche, Zheng, and Mengshoel 2011; Lerner, Parr, Koller, and Biswas 2000; Rose, Smaili, and Charpillet 2005).

Just as BNs provide a compact representation of a joint probability distribution, DBNs can additionally encode the evolution of a dynamic joint probability distribution, thus allowing predictions about future states of a system.

We propose a novel monitoring approach that combines the power of ontologies with DBNs. In a nutshell, we use DBNs to define a state-transition system over a distinguished and fixed set of variables to be monitored, and ontologies to encode the rich background knowledge. Considering a finite, fixed time horizon, as opposed to temporal DLs, we provide a succinct querying mechanism based on linear temporal logic (Pnueli 1977) and define reasoning problems, which generalize classical inference problems of DBNs towards an ontological setting, all of which are to the best of our knowledge, first of its kind, in a ontological setting.

8.2.1 Dynamic Bayesian Networks

BNs are static models, i.e., it is not possible to capture the dynamic features of the application domain. For instance, the probability of a patient having a high fever is very likely given the fact that the patient had high fever in the previous time step. Such scenarios can be modeled using dynamic Bayesian networks (DBNs) (Dean and Kanazawa 1989; Murphy 2002), which extend BNs to provide a compact representation of evolving JPDs for a fixed set of random variables.

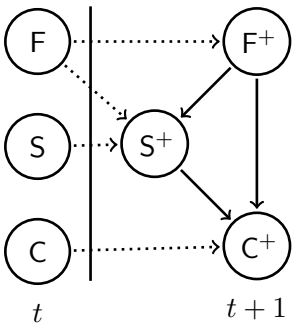
BNs are known for compactly representing a state space, while DBNs can also represent the state-transition probabilities, and thus, can be facilitated to make projections about the future states of a system. The update of the joint probability distribution is typically expressed through an two-slice BN, which expresses the probabilities at the next point in time, given the current context.

Formally, a *two-slice BN (TBN)* over a finite set of variables V is a pair $\mathcal{B}_{\to} = (G, \Theta)$, where $G = (V \cup V^+, E)$ with $V^+ = \{X^+ \mid X \in V\}$ is a DAG such that all edges are directed from elements of $V \cup V^+$ to elements of V^+, and Θ contains, for every $X^+ \in V^+$, a conditional probability distribution $P(X^+ \mid \pi(X^+))$ for X^+ given the parents of X^+. As standard in BNs, every node is independent of all its non-descendants given its parents in TBNs. Thus, for a TBN $\mathcal{B}_{\to}$, the conditional probability distribution at time $t+1$ given time t is

$$P_{\mathcal{B}_{\to}}(V_{t+1} \mid V_t) = \prod_{X^+ \in V^+} P_{\mathcal{B}_{\to}}(X^+ \mid \pi(X^+)).$$

Example 8.26 Figure 8.2 depicts a TBN $\mathcal{B}_{\to}$ and thereby defines the transition probabilities between the two time slices. For instance, the probability for bob to have high fever at time $t+1$ provided he did not have high fever at time t is given by $\mathrm{P}_{\mathcal{B}_{\to}}(\mathsf{f}^+ \mid \neg\mathsf{f}) = 0.1$. ◊

A *dynamic Bayesian network (DBN)* is a pair $\mathcal{D} = (\mathcal{B}_1, \mathcal{B}_{\to})$, where $\mathcal{B}_1$ is a BN , and $\mathcal{B}_{\to}$ is a TBN, defined over V. Intuitively, a DBN's set of nodes of the graph can be



	f^+
f	.7
$\neg\mathsf{f}$	.1

			s^+
f	s	f^+	.9
f	s	$\neg\mathsf{f}^+$	.5
f	$\neg\mathsf{s}$	f^+	.8
f	$\neg\mathsf{s}$	$\neg\mathsf{f}^+$	.4
$\neg\mathsf{f}$	s	f^+	.8
$\neg\mathsf{f}$	s	$\neg\mathsf{f}^+$	.4
$\neg\mathsf{f}$	$\neg\mathsf{s}$	f^+	.7
$\neg\mathsf{f}$	$\neg\mathsf{s}$	$\neg\mathsf{f}^+$	.1

			c^+
c	f^+	s^+	.7
c	f^+	$\neg\mathsf{s}^+$	.2
c	$\neg\mathsf{f}^+$	s^+	.4
c	$\neg\mathsf{f}^+$	$\neg\mathsf{s}^+$	1
$\neg\mathsf{c}$	f^+	s^+	.6
$\neg\mathsf{c}$	f^+	$\neg\mathsf{s}^+$	.1
$\neg\mathsf{c}$	$\neg\mathsf{f}^+$	s^+	.3
$\neg\mathsf{c}$	$\neg\mathsf{f}^+$	$\neg\mathsf{s}^+$	1

Figure 8.2: The DBN $\mathcal{D}_h = (\mathcal{B}_1, \mathcal{B}_\to)$, consisting of (a) a BN $\mathcal{B}_1$ (= $\mathcal{B}_h$) over $V = \{\mathsf{F}, \mathsf{S}, \mathsf{C}\}$, which compactly represents a joint probability distribution, and (b) a two-slice BN $\mathcal{B}_\to$ over V, which defines the transition probabilities between two time slices V_t and V_{t+1}.

thought of as containing two disjoint copies of the random variables in V, where the probability distribution at time $t+1$ depends on the distribution at time t.

To be able to distinguish the variables in different time slices, we use V_t and X_t to denote the set of variables V and the variable $X \in V$ at time t, respectively. As in BNs, x is an abbreviation for $X = 1$ and $\neg x$ for $X = 0$. Moreover, we assume the (first-order) Markov property: the probability of the future state is independent from the past, given the present state. We note, however, that all of our results can be generalized to k-slice BNs, which relaxes this assumption to k slices and adds memory. Given the (first-order) Markov property, a DBN $\mathcal{D} = (\mathcal{B}_1, \mathcal{B}_\to)$ defines, for every $t \geq 1$, the unique joint probability distribution

$$\mathrm{P}_\mathcal{D}(V_t) = \mathrm{P}_{\mathcal{B}_1}(V_1) \cdot \prod_{i=2}^{t} \prod_{X_i \in V_i} \mathrm{P}_{\mathcal{B}_\to}(X_i \mid \pi(X_i)).$$

We briefly illustrate these notions on our running example.

Example 8.27 Consider the TBN $\mathcal{B}_\to$ depicted in Figure 8.2. The pair $\mathcal{D} = (\mathcal{B}_1, \mathcal{B}_\to)$ is a DBN, where $\mathcal{B}_1$ is the BN depicted in Figure 8.1. We can pose non-statics queries to the DBN $\mathcal{D}$. For instance, the probability of bob having high fever at time point 2 , $\mathrm{P}_{\mathcal{D}_h}(\mathsf{f}_2)$, can be computed as

$$\mathrm{P}_{\mathcal{B}_1}(\mathsf{f}_1) \cdot \mathrm{P}_{\mathcal{B}_\to}(\mathsf{f}_2 \mid \mathsf{f}_1) + \mathrm{P}_{\mathcal{B}_1}(\neg\mathsf{f}_1) \cdot \mathrm{P}_{\mathcal{B}_\to}(\mathsf{f}_2 \mid \neg\mathsf{f}_1),$$

which is a dynamic version of standard probabilistic inference of BNs . ◊

Intuitively, the distribution at time t is defined by unraveling the DBN starting from $\mathcal{B}_1$, using the two-slice structure of $\mathcal{B}_\to$ until t copies of V have been created. This produces a new BN $\mathcal{B}_{1:t}$ encoding the distribution over time of the different variables. Figure 8.3 shows the unraveling to $t = 3$ of the DBN $(\mathcal{B}_1, \mathcal{B}_\to)$, where $\mathcal{B}_1$ and $\mathcal{B}_\to$ are the networks shown in Figures 8.1 and 8.2, respectively.

The conditional probability tables of each node given its parents (not shown) are those of $\mathcal{B}_1$ for the nodes in V_1, and of $\mathcal{B}_\to$ for nodes in $V_2 \cup V_3$. Notice that $\mathcal{B}_{1:t}$ has

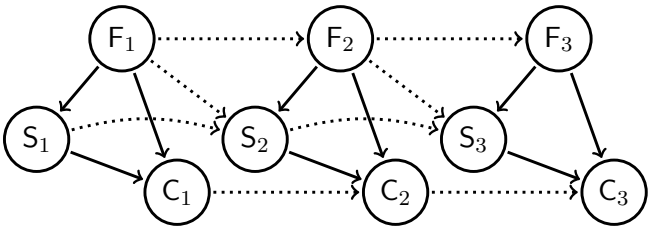


Figure 8.3: Three step unraveling $\mathcal{B}_{1:3}$ of $(\mathcal{B}_1, \mathcal{B}_{\rightarrow})$

t copies of each random variable in V. For a given $t \geq 1$, we call $\mathcal{B}_t$ the BN obtained from the unraveling $\mathcal{B}_{1:t}$ of the DBN to time t, and eliminating all variables not in V_t. In particular, we have that $P_{\mathcal{B}_t}(V) = P_{\mathcal{B}_{1:t}}(V_t)$.

For notational convenience, we write $V_{1:t} := V_1 \cup \ldots \cup V_t$, and $\mathcal{W}_{1:t}$ for a valuation of $V_{1:t}$. Moreover, we write $\mathbf{X}_t \subseteq V_t$ to denote a *set of variables* at time t and $\mathbf{x}_t$ to denote an *instantiation of these variables*; if, furthermore, $\mathbf{X}_t = V_t$ then $\mathbf{x}_t$ corresponds to a state at time t. Analogously, we write $\mathbf{X}_{1:t}$ to denote a sequence of variables $x_1 \ldots x_t$ and $\mathbf{x}_{1:t}$ to denote an instantiation of these variables, which is usually called a *trajectory*.

Traditional inference problems in DBNs are given in Figure 8.4 with the help of a simple timeline. Formally, given a DBN, *filtering (also called monitoring)* is the task of computing $\mathrm{P}_{\mathcal{D}}(\mathbf{x}_t \mid \mathbf{y}_{1:t})$. *Smoothing* and *prediction* are the past ($\mathrm{P}_{\mathcal{D}}(\mathbf{x}_{t-l} \mid \mathbf{y}_{1:t})$) and future ($\mathrm{P}_{\mathcal{D}}(\mathbf{x}_{t+h} \mid \mathbf{y}_{1:t})$) analogs of filtering, respectively; and *classification* is a special case of filtering, which amounts to computing $\mathrm{P}_{\mathcal{D}}(\mathbf{y}_{1:t})$. Finally, finding a hypothesis, which maximizes the probability of the query is called *decoding*, and is computed as $\arg\max_{\mathbf{x}_{1:t}} \mathrm{P}_{\mathcal{D}}(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$. Decoding is analogous to *Maximum a Posteriori Hypothesis (MAP)* and *most probable explanation (MPE)* in BNs (Park and Darwiche 2004b). For further details, we refer to (Murphy 2002).

8.2.2 Syntax and Semantics of Dynamic Bayesian Ontology Languages

The syntax of dynamic Bayesian Ontology Languages is identical to BOLs except the fact that DBOLs allow a DBN instead of a BN.

Definition 8.28 (dynamic KB) A *dynamic KB* is a tuple $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{D})$ where $\mathcal{D}$ is a DBN, $\mathcal{T}$ is a probabilistic TBox (over the DBN $\mathcal{D}$), and $\mathcal{A}$ a probabilistic ABox (over the DBN $\mathcal{D}$). ◊

Within the scope of this paper, we limit ourselves to a *fixed, finite* horizon. In other words, all traces are assumed to be over a fixed, finite bound. We will denote the bound with $\theta \in \mathbb{N}$. By this assumption, dynamic KBs can be viewed as *time-stamped* probabilistic KBs. More formally, given a vocabulary of probabilistic KBs, which consists of concept ($\mathsf{N_C}$), role ($\mathsf{N_R}$), individual ($\mathsf{N_I}$) and variable (V) names, we define the extended vocabulary of *time-stamped* concept ($\mathsf{N_C}_i$), role ($\mathsf{N_R}_i$), individual ($\mathsf{N_I}_i$) and variable (V_i) names, respectively, for $1 \leq i \leq \theta$. Intuitively, we consider copies of the KB, that makes the temporal information explicit. Thus, for a given a dynamic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{D})$, we write $\mathcal{T}_i$ (resp., $\mathcal{A}_i$) to denote the copy of $\mathcal{T}$ (resp., $\mathcal{A}$) over the extended vocabulary. The semantics of dynamic KBs are given in terms of temporal interpretations.

Figure 8.4: A timeline with a reference time point t. *Filtering* is the task of computing $\mathrm{P}_{\mathcal{D}}(\mathbf{x}_t \mid \mathbf{y}_{1:t})$; *smoothing* and *prediction* are the past $(\mathrm{P}_{\mathcal{D}}(\mathbf{x}_{t-l} \mid \mathbf{y}_{1:t}))$ and future $(\mathrm{P}_{\mathcal{D}}(\mathbf{x}_{t+h} \mid \mathbf{y}_{1:t}))$ analogs of filtering, respectively; and *classification* is a special case of filtering, which amounts to computing $\mathrm{P}_{\mathcal{D}}(\mathbf{y}_{1:t})$. *Decoding* corresponds to computing $\arg\max_{\mathbf{x}_{1:t}} \mathrm{P}_{\mathcal{D}}(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$.

Definition 8.29 (temporal interpretations) Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{D})$ be a dynamic KB. A *temporal interpretation* $\mathbb{I}$ is a finite sequence $(\mathcal{I})_{t>0}$ of interpretations; $\mathbb{I}$ satisfies $(\mathcal{T}, \mathcal{A})$ if and only if $\mathcal{I}_t \models (\mathcal{T}_t, \mathcal{A}_t)$ for all $t > 0$. A *probabilistic temporal interpretation* is a pair $\mathbf{I} = (\mathfrak{I}, \mathrm{P}_{\mathfrak{I}})$, where $\mathfrak{I}$ is a set of temporal interpretations $\mathbb{I}$, and $\mathrm{P}_{\mathfrak{I}}$ is a probability distribution over $\mathfrak{I}$ such that $\mathrm{P}_{\mathfrak{I}}(\mathbb{I}) > 0$ only for finitely many interpretations $\mathbb{I} \in \mathfrak{I}$. $\mathbf{I}$ is a *model* of the TBox $\mathcal{T}$ (resp., ABox $\mathcal{A}$) if every $\mathbb{I} \in \mathfrak{I}$ satisfies $\mathcal{T}$ (resp., $\mathcal{A}$). $\mathbf{I}$ is *consistent* with the DBN $\mathcal{D}$ if for every valuation $\mathcal{W}_{1:t}$ of (finitely many) variables in $V_{1:t}$ it holds that

$$\sum_{\mathbb{I} \in \mathfrak{I}, \mathcal{V}^{\mathbb{I}}_{1:t} = \mathcal{W}_{1:t}} \mathrm{P}_{\mathfrak{I}}(\mathbb{I}) = \mathrm{P}_{\mathcal{B}_{1:t}}(\mathcal{W}_{1:t}).$$

The probabilistic temporal interpretation $\mathbf{I}$ is a *model* of the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{D})$ if and only if it is a model of $(\mathcal{T}, \mathcal{A})$, and it is consistent with the DBN $\mathcal{D}$. ◊

The intuition behind dynamic Bayesian ontology languages is very simple. Recall that in BOLs, the ontological knowledge is assumed to hold within a certain world and each such world has a probability defined by the BN. By replacing the BN with a DBN, DBOLs can now also encode transitions between worlds/states which triggers changes in the ontological knowledge. In other words, DBNs define a state-transition system over a set of distinguished variables and each state of the system is equipped with ontological knowledge that is required to hold in the given state. This framework gives rise to advanced querying mechanisms, as we elaborate next.

8.2.3 Ontological Monitoring

DBNs encode a the dynamic evolution of a system but while querying they are rather limited as pointed before in (Langmead 2009). Briefly, inference in DBNs require to explicitly specify the time points of the events. On the other hand, the dynamic evolution shows a logical structure and an adequate query language should exploit these structures. This limitation is addressed in (Langmead 2009) by introducing a time-bounded query language based on LTL (Pnueli 1977); thus compactly encoding temporal logical relations. Clearly, the temporal queries are superior to standard queries and they allow specifying complex temporal relations which would not be possible otherwise.

Similarly, in dynamic Bayesian ontologies, we are interested in posing complex temporal queries. For example, a simple query in a health-care context that asks the probability of some patient eventually entering into a critical situation is hard to formulate without

temporal constructs. In a similar spirit to (Langmead 2009), we adopt a temporal query language to capture such complex logical relations. Differently, however, our query language is defined over CQs since we now have a first-order representation.

Several temporal extensions of conjunctive queries exist. We concentrate on an extension, based the well-known temporal logic LTL (Pnueli 1977). Inspired by temporal conjunctive queries (TCQs), first defined in (Baader, Borgwardt, and Lippmann 2015), we define a query language by allowing the use of CQs in place of propositional variables.

Definition 8.30 (query language) A (positive) TCQ is inductively defined by the grammar rule[1]

$$Q ::= Q \wedge Q \mid Q \vee Q \mid \bigcirc Q \mid \bigcirc^{-} Q \mid \Diamond Q \mid \Box Q \mid Q \mathsf{U} Q \mid Q \mathsf{S} Q,$$

where Q is a CQ. The constructors denote *conjunction*, *disjunction*, *next*, *previous*, *eventually*, *always*, *until* and *since*, respectively. A TCQ is *bounded* if the temporal operators can only range over a fixed, finite bound θ. ◊

As stated earlier, in this paper, we always assume a fixed bound on the temporal operators. Notice that this makes our query language simpler than LTL, which can range over infinite traces, in general. However, LTL over finite (but unbounded) traces has also been studied in the literature (De Giacomo and Vardi 2013). As we will see later, TCQs can be unraveled to unions of conjunctive queries, and this unraveling may be of size exponential; thus, TCQs serve as a *succinct* way of specifying (bounded) dynamic queries. We first define the temporal query semantics in the classical way.

Definition 8.31 (query semantics) Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{D})$ be a dynamic KB. The (θ-bounded) query semantics w.r.t. a temporal interpretation $\mathbb{I}$ is inductively defined as follows.

$$
\begin{array}{lll}
(\mathbb{I}, t) \models Q_1 & \textit{iff} & \mathcal{I}_t \models Q_1 \\
(\mathbb{I}, t) \models Q_1 \wedge Q_2 & \textit{iff} & (\mathbb{I}, t) \models Q_1 \textit{ and } (\mathbb{I}, t) \models Q_2 \\
(\mathbb{I}, t) \models Q_1 \vee Q_2 & \textit{iff} & (\mathbb{I}, t) \models Q_1 \textit{ or } (\mathbb{I}, t) \models Q_2 \\
(\mathbb{I}, t) \models \bigcirc Q_1 & \textit{iff} & \theta > t \text{ and } (\mathbb{I}, t+1) \models Q_1 \\
(\mathbb{I}, t) \models \bigcirc^{-} Q_1 & \textit{iff} & t > 1 \text{ and } (\mathbb{I}, t-1) \models Q_1 \\
(\mathbb{I}, t) \models \Diamond Q_1 & \textit{iff} & \exists i \in [t, \theta] : \mathcal{I}_i \models Q_1 \\
(\mathbb{I}, t) \models \Box Q_1 & \textit{iff} & \forall i \in [t, \theta] : \mathcal{I}_i \models Q_1 \\
(\mathbb{I}, t) \models Q_1 \mathsf{U} Q_2 & \textit{iff} & \exists k \in [t, \theta] \textit{ such that } \forall\, \mathcal{I}_i \in \mathbb{I}, t \leq i < k, \mathcal{I}_i \models Q_1 \textit{ and } \mathcal{I}_k \models Q_2 \\
(\mathbb{I}, t) \models Q_1 \mathsf{S} Q_2 & \textit{iff} & \exists j \in [1, t] \textit{ such that } \forall\, \mathcal{I}_i \in \mathbb{I}, j \leq i < t, \mathcal{I}_i \models Q_1 \textit{ and } \mathcal{I}_j \models Q_2
\end{array}
$$

Finally, $(\mathcal{A}, t) \models (\mathcal{T}, Q)$ if and only if for all temporal interpretations $(\mathbb{I}, t) \models (\mathcal{T}, \mathcal{A})$ it holds that $(\mathbb{I}, t) \models Q$. ◊

The query semantics can be seen as an adaptation of the semantics of TCQs (Baader, Borgwardt, and Lippmann 2015) restricted to a fixed bound. We are interested in probabilistic reasoning in dynamic Bayesian ontology languages, which we formalize next.

[1] Notice that **true** is in the query language (empty CQ) and thereby $\Diamond_n Q$ is only an abbreviation of the formula **true** $\mathsf{U}\ \varphi$.

Definition 8.32 (ontological monitoring) Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{D})$ be a probabilistic KB. The probability of a Boolean TCQ Q w.r.t. a probabilistic temporal interpretation $\mathbf{I} = (\mathfrak{I}, \mathrm{P}_{\mathfrak{I}})$ at time $1 \leq t < \theta$ is given by

$$\mathrm{P}_{\mathbf{I}}(Q, t) \coloneqq \sum_{\mathbb{I} \in \mathfrak{I}, (\mathbb{I}, t) \models Q} \mathrm{P}_{\mathfrak{I}}(\mathbb{I})$$

The *probability* of a TCQ Q at time $1 \leq t < \theta$ w.r.t. $\mathcal{K}$, denoted $\mathrm{P}_{\mathcal{K}}(Q, t)$, is given by

$$\mathrm{P}_{\mathcal{K}}(Q, t) \coloneqq \inf_{\mathbf{I} \models \mathcal{K}} \mathrm{P}_{\mathbf{I}}(Q, t).$$

Given a probabilistic KB $\mathcal{K}$, a Boolean TCQ Q, and a threshold value $p \in [0, 1)$, *ontological monitoring* is to decide whether $\mathrm{P}_{\mathcal{K}}(Q, t) > p$. ◊

We briefly illustrate ontological monitoring on our running example.

Example 8.33 Consider the dynamic KB $\mathcal{K} = (\mathcal{T}_h, \mathcal{A}_h, \mathcal{D}_h)$, where $\mathcal{D}_h = (\mathcal{B}_h, \mathcal{B}_{\rightarrow})$. Then, by posing the query

$$Q_1 \coloneqq \Diamond(Q_{\mathsf{bob}} = \exists x\ \mathsf{Patient}(\mathsf{bob}) \wedge \mathsf{risk}(\mathsf{bob}, x) \wedge \mathsf{CriticalSituation}(x)),$$

we can compute the probability of bob eventually going into critical situation. The query

$$Q_2 \coloneqq \bigcirc^{-}(Q'_{\mathsf{bob}} = \exists x\ \mathsf{finding}(\mathsf{bob}, x) \wedge \mathsf{Hypertension}(x)),$$

asks whether bob has hypertension in the previous time point. We can, of course, form much more elaborate queries such as $(Q_1 \mathsf{U} Q_2)$. ◊

As we are only concerned with a bounded-horizon, we can always unravel the dynamic knowledge base into a static one in a similar way that the DBN can be unraveled into a BN. More precisely, given a dynamic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{D})$ where $\mathcal{D} = (\mathcal{B}_1, \mathcal{B}_{\rightarrow})$, we define

$$\mathcal{T}_{1:\theta} = \bigcup_{0 < i \leq \theta} \mathcal{T}_i \quad \textit{and} \quad \mathcal{A}_{1:\theta} = \bigcup_{0 < i \leq \theta} \mathcal{A}_i$$

as the unravellings of $\mathcal{T}$ and $\mathcal{A}$, respectively. Then, the *unraveling* of the dynamic KB $\mathcal{K}$ is the probabilistic KB $\mathcal{K}_{1:\theta} = (\mathcal{T}_{1:\theta}, \mathcal{A}_{1:\theta}, \mathcal{B}_{1:\theta})$ where $\mathcal{B}_{1:\theta}$ is defined as before. The abstractions to worlds are defined analogously: the KB $\mathcal{K}_{\mathcal{W}_{1:\theta}} = (\mathcal{T}_{\mathcal{W}_{1:\theta}}, \mathcal{A}_{\mathcal{W}_{1:\theta}})$ represents the restriction of $\mathcal{K}$ to a valuation $\mathcal{W}_{1:\theta}$ of the variables $V_{1:\theta}$.

Furthermore, TCQs with bounded horizons can also be unraveled to static queries with respect a relevant time point t. In particular, since we do not allow negation, it is easy to see that such unraveling necessarily produces unions of conjunctive queries. Similarly, the atoms of the resulting UCQ have to be renamed to time-stamped atoms to preserve the query semantics. Let us denote the unraveling of a query Q with respect to t by $Q_{[1:\theta]}$. These observations entail the following result.

Theorem 8.34 *Given a dynamic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{B})$ and a Boolean TCQ Q, it holds that*

$$(\mathcal{A}, t) \models (\mathcal{T}, Q) \textit{ if and only if } \mathcal{A}_{\mathcal{W}_{1:\theta}} \models (\mathcal{T}_{\mathcal{W}_{1:\theta}}, Q_{1:\theta})$$

where $\mathcal{A}_{\mathcal{W}_{1:\theta}}$ is a classical ABox, $\mathcal{T}_{\mathcal{W}_{1:\theta}}$ is a classical TBox and $Q_{1:\theta}$ is a UCQ.

Proof. Given the dynamic KB $(\mathcal{T}, \mathcal{A}, \mathcal{D})$, we consider the unraveling $(\mathcal{T}_{\mathcal{W}_{1:\theta}}, \mathcal{A}_{\mathcal{W}_{1:\theta}}, \mathcal{B}_{1:\theta})$ over time-stamped copies of the original vocabulary. Suppose that $\mathbb{I} \models (\mathcal{T}, \mathcal{A})$ holds for some timed interpretation $\mathbb{I} = (\mathcal{I})_{t>0}$. Let us denote the copy of the interpretation $\mathcal{I}_t$ over the vocabulary marked with t by $\mathcal{I}_{c_t} = (\Delta^{\mathcal{I}}_t, \cdot^{\mathcal{I}}_t, \mathcal{V}^{\mathcal{I}}_t)$. We then define the interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}}, \mathcal{V}^{\mathcal{J}})$ where

$$\Delta^{\mathcal{J}} = \bigcup_{0<i\leq\theta} \Delta^{\mathcal{I}}_t, \quad \cdot^{\mathcal{J}} = \bigcup_{0<i\leq\theta} \cdot^{\mathcal{I}}_t, \quad \mathcal{V}^{\mathcal{J}} = \bigcup_{0<i\leq\theta} \mathcal{V}^{\mathcal{I}}_t.$$

It is easy to see that $\mathbb{I} \models (\mathcal{T}, \mathcal{A})$ if and only if $\mathcal{J} \models (\mathcal{T}_{\mathcal{W}_{1:\theta}}, \mathcal{A}_{\mathcal{W}_{1:\theta}})$ and that $(\mathbb{I}, t) \models Q$ if and only if $\mathcal{J} \models Q_{1:\theta}$, by the given construction.

Conversely, consider the unraveling $(\mathcal{T}_{\mathcal{W}_{1:\theta}}, \mathcal{A}_{\mathcal{W}_{1:\theta}}, \mathcal{B}_{1:\theta})$ and assume now that there exists a model $\mathcal{I}$ of $(\mathcal{T}_{\mathcal{W}_{1:\theta}}, \mathcal{A}_{\mathcal{W}_{1:\theta}})$. We define the interpretation $\mathbb{I} = (\mathcal{J})_{t>0}$ where every interpretation $\mathcal{J}_t$ corresponds to the restriction of the interpretation $\mathcal{J}$ to time t modulo renaming. As before, it is easy to check $\mathbb{I} \models (\mathcal{T}, \mathcal{A})$ and $\mathcal{J} \models Q_{1:\theta}$ if and only if $(\mathbb{I}, t) \models Q$ where $Q_{1:\theta}$ is unraveled with respect to t. □

Intuitively, TCQs can be seen as a compact encoding of time-stamped unions of conjunctive queries. In fact, the unraveling can produce a UCQ that is of size exponential in the nesting size of temporal quantifiers. With the help of Theorem 8.34, we easily obtain a reduction from ontological monitoring to Bayesian query evaluation.

Corollary 8.35 *Given a dynamic KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{D})$ and a TCQ Q, it holds that*

$$\mathrm{P}_{\mathcal{K}}(Q, t) = \sum_{\mathcal{A}_{\mathcal{W}_{1:\theta}} \models (\mathcal{T}_{\mathcal{W}_{1:\theta}}, Q_{1:\theta})} \mathrm{P}_{\mathcal{B}_{1:\theta}}(\mathcal{W}_{1:\theta}) = \mathrm{P}_{\mathcal{K}_{1:\theta}}(Q_{1:\theta})$$

where $\mathcal{W}_{1:\theta}$ denotes a valuation over the variables $V_{1:\theta}$ of the BN $\mathcal{B}_{1:\theta}$ (unraveling of $\mathcal{D}$), $\mathcal{A}_{\mathcal{W}_{1:\theta}}$ is a classical ABox, $\mathcal{T}_{\mathcal{W}_{1:\theta}}$ is a classical TBox and $Q_{1:\theta}$ is a UCQ.

Given the fixed-bound assumption, it is easy to see that the unraveling of the dynamic knowledge base is linear in the size of θ. Since the query is assumed to be fixed in data complexity, Theorem 8.34 and Corollary 8.35 immediately entail the following results.

Corollary 8.36 *For all DLs $\mathcal{L}$, ontological monitoring can be polynomially reduced to Bayesian query evaluation in data complexity.*

As a consequence, all data complexity results for Bayesian query evaluation apply to ontological monitoring. Note that for the combined complexity, there is no immediate reduction. Intuitively, while expanding the temporal operators, such as until and since, we have choices for the durations of these operators. Therefore, the naïve approach given in Theorem 8.34 would result in an exponential blow-up. Pinpointing the precise complexity of ontological monitoring is left as a future work.

As before, it is desirable to incorporate evidence into queries as in Bayesian query evaluation. Instead of the decision problem $\mathrm{P}_{\mathcal{K}}(Q, t) > p$, we define $\mathrm{P}_{\mathcal{K}}(Q, t \mid \mathbf{e}_{1:\theta}) > p$, as before, with the only difference that $\mathbf{e}_{1:\theta}$ is now over the variables $V_{1:\theta}$. Observe that queries of this form generalize classical inference tasks in DBNs. Filtering, smoothing and prediction are all special cases of such queries. The only inference task which cannot be captured with temporal queries is decoding, which is closely related to the problem of

most likely context. Given the Corollary 8.36, it is easy to see that most likely context can be generalized towards *ontological decoding*. Investigating such connections in more detail is also part of future work.

8.3 Related Work and Outlook

Bayesian ontology languages extends the underlying ontology language with a propositional abstraction given in terms of contexts. BOLs represent certain knowledge that is dependent on an uncertain context. The uncertainty of the different contexts is expressed by a probability distribution represented via a Bayesian network. We investigated query answering in Bayesian ontology languages along with other reasoning tasks.

Combining probabilistic graphical models with terminological knowledge is an old idea that dates back to P-Classic (Koller, Levy, and Pfeffer 1997) which extended Classic through probability distributions over the interpretation domain. A more recent work PR-OWL (Costa, K. B. Laskey, and K. J. Laskey 2008) uses multi-entity BNs to describe the probability distributions of some domain elements. In both cases, the probabilistic component is interpreted providing individuals with a probability distribution; this differs greatly from the possible world semantics, in which we consider a probability distribution over a set of classical DL interpretations.

Perhaps the closest to our approach are the Bayesian extension of *DL-Lite* (d'Amato, Fanizzi, and Lukasiewicz 2008) and DISPONTE (Riguzzi, Bellodi, Lamma, and Zese 2015). The latter allows for so-called epistemic probabilities that express the uncertainty associated to a given axiom. Their semantics are based, as ours, on a probabilistic distribution over a set of interpretations. The main difference with our approach is that in (Riguzzi et al. 2015), the authors assume that all probabilities are independent, while we provide a joint probability distribution through the BN. Note also that a major advantage of BOLs is the re-usability of the context variables for different axioms as already pointed out.

The Bayesian Description Logic given in (d'Amato, Fanizzi, and Lukasiewicz 2008) is based on *DL-Lite* and looks almost identical to ours. There is, however, a subtle but important difference. In our approach, an interpretation $\mathcal{I}$ satisfies an axiom $\langle C \sqsubseteq D : \varphi \rangle$ if $\mathcal{V}^{\mathcal{I}} \models \varphi$ implies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. In (d'Amato, Fanizzi, and Lukasiewicz 2008), the authors employ a closed-world assumption over the contexts, where this implication is substituted for an equivalence; i.e., $\mathcal{V}^{\mathcal{I}} \models \varphi$ also implies $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$. The use of such semantics can easily produce inconsistent KBs, which is highly undesirable.

The DL $\mathcal{EL}$ and *DL-Lite* are also investigated in the context of ontology based access to probabilistic data (Jung and Lutz 2012), which we already reviewed in Chapter 7. Recently, approaches using discrete probability distributions over (database) facts are extended to the case of continuous probability distributions over numerical values (Baader, Koopmann, and Turhan 2017). There exists a plethora of different probabilistic DLs tailored for different applications. For a slightly outdated survey on probabilistic DLs, we refer the reader to (Lukasiewicz and Straccia 2008).

From a broader perspective, our work relates to (lifted) probabilistic graphical models such as MLNs (Richardson and Domingos 2006), or relational Bayesian networks (Jaeger 1997). This line of research is also motivated by the limitations of propositional models.

Their semantics is, however, given over the groundings, i.e., over a fixed set of constants, which differs greatly with the semantics of ontology languages.

To the best of our knowledge dynamic Bayesian ontology languages is the first proposal in the DL community that combined dynamic and probabilistic reasoning in order to do probabilistic monitoring based on ontologies. It is well-known that DBNs generalize hidden Markov models and Kalman filters as they are factored representation of these models. Factored representations of Markov Decision Processes has been widely studied (Degris and Sigaud 2010). Importantly, the complexity of solving Markov Decision Problems turns out to be very different in factored representations (Littman, Dean, and Kaelbling 1995; Mundhenk, Goldsmith, Lusena, and Allender 2000). We also note that it is common to make the fixed-horizon assumption, see for instance the complexity analysis on partially observable Markov decision processes (Mundhenk et al. 2000).

Part IV

Conclusions

Chapter 9

Conclusions

As a conclusion to this thesis, we first summarize the obtained results about query answering in probabilistic data and knowledge bases and then discuss future research directions.

9.1 Summary and Outlook

The motivation behind marrying logic and probability is to combine the capacity of probability theory to handle pervasive uncertainty with the capacity of mathematical logic to exploit the structure of formal argument. Recent milestones in combining logic and probability was on the one hand through the notion of weighted model counting and on the other hand through the shift from propositional probabilistic models to first-order probabilistic models in AI. Our research goal in this thesis is to combine first-order models and probabilistic models in the context of large scale probabilistic knowledge bases and to enhance large-scale knowledge bases with more realistic data models, thereby allowing for better means for querying them.

Our main contributions can be summarized as follows. We developed different rigorous semantics for probabilistic data and knowledge bases, analyzed their computational properties relative to a wide range of query languages and identified sources of in/tractability and designed practical scalable query answering algorithms whenever possible. We now provide a more detailed overview of the results.

Overview of Part II: Probabilistic Databases. This part only covers the results on Probabilistic Databases and is organized in three chapters. In essence, this part serves as the basis for the results presented in Part III.

Chapter 3. In this chapter, we deepened the research on probabilistic query evaluation in classical PDBs, which also serve as a baseline of the remaining of the thesis. We followed a decision-theoretic approach and defined probabilistic query evaluation as a decision problem and studied its computational complexity. We have shown that the data complexity dichotomy of the computation problem transfers to the decision problem under polynomial time Turing reductions. Beyond the known results, we also obtained combined complexity results.

Chapter 4. In this chapter, we provided a deep discussion regarding the deficiencies of PDB semantics, which led to the proposal of *open-world probabilistic databases*. We argued in detail, and on several concrete examples how OpenPDB semantics improves

on the PDB semantics with respect to the goals identified in this work. We have shown that the *data complexity dichotomy* for unions of conjunctive queries in PDBs (Dalvi and Suciu 2012) can be lifted to OpenPDBs. Moreover, the class of safe queries is supported by a special lifted inference algorithm, which performs first-order manipulations and avoids explicit grounding. As a side contribution, we noted that the safe queries in PDBs can actually be computed in *linear time* under reasonable assumptions. For queries that involve negation, we have shown that inference becomes harder for OpenPDBs than for PDBs already in data complexity. In a technically involved result, we show that this can is the case even for a safe query.

Chapter 5. In this chapter, we studied two alternative inference tasks for probabilistic databases; namely finding the *most probable database* and the *most probable hypothesis* for a given query, both of which are inspired by the *maximal posterior probability* computations of Probabilistic Graphical Models. The main contributions are the complexity results obtained for these problems. Notably, all the results are completeness results except for TC^0 upper bounds, which is closely related to deciding the most significant bit of multiplication.

Overview of Part III: Logic and Probabilistic Knowledge Bases. This part extended the results from Part II to also incorporate commonsense knowledge in the form of ontologies. The most prominent ontology languages are based on Description Logics and Datalog$^{\pm}$; these are reviewed in Chapter 6 together with ontology-mediated query answering. The contributions are given in the next chapters.

Chapter 7. In this chapter, we extended the problems introduced in Part II to the case of ontology-mediated queries based on Datalog$^{\pm}$. First, we studied probabilistic ontology-mediated query evaluation for PDBs. We have lifted the data complexity dichotomy from Part I also to probabilistic ontology-mediated query evaluation for FO-rewritable Datalog$^{\pm}$ languages (the case for languages with P-complete data complexity for OMQA remains open). We have made a thorough complexity analysis with different complexity-theoretic assumptions and provided a complete picture in terms of computational complexity. Analogously, we investigated probabilistic ontology-mediated query evaluation for OpenPDBs and lifted the data complexity dichotomy result in OpenPDBs to *positive* FO-rewritable Datalog$^{\pm}$ languages (while the general case remains open). Finally, the most probable database and most probable hypothesis problems are revisited in the context of ontology-mediated queries. In each of these sections, we provided a host of complexity results and obtained a mostly complete picture.

Chapter 8. The focus of this chapter is on Bayesian ontology languages that extend classical Description Logics with the capability of representing and reasoning over uncertain domains. In the first section, we study ontology-mediated query evaluation in Bayesian ontology languages together with two other reasoning problems, called *most likely context* and *most likely world*. These problems are supported with a thorough complexity analysis. In the second part of this chapter, we propose a novel monitoring approach that combines the power of ontology languages with *dynamic* BNs. The

resulting formalism is then called dynamic Bayesian ontology languages and allows to make projections about the future states of a system. To the best of our knowledge dynamic Bayesian ontology languages is the first of its kind in Description Logics. Under the fixed-horizon assumption, we show that monitoring tasks in dynamic Bayesian ontology languages can be reduced to probabilistic query evaluation in (static) Bayesian ontology languages.

9.2 Future Research

Querying probabilistic databases is extremely demanding from a computational point of view. Developing scalable query answering algorithms is therefore a big challenge and even more so for more powerful data models. The obvious question is, of course, how to achieve substantial scalability gains for query answering over realistic data models. A possible road-map is to make further progress on recent techniques; namely, to develop scalable (1) lifted, (2) special-case, and (3) approximate query answering algorithms, as well as novel scalable compilation techniques, (4) generalizing existing compilation techniques, (5) based on recent progress in tensor factorization, and (6) inspired by deep learning.

Another major challenge is to *classify the landscape of tractable query answering.* This includes finding out which kinds of queries can be evaluated in polynomial time over which kinds of commonsense knowledge, and looking for a dichotomy between easy and hard queries. Although the complexity study we pursued in this work resulted in a mostly complete picture, this is not the case for classification results. For instance, it is open whether a data complexity dichotomy can be obtained in PDBs for queries that contain negation in front of the query atoms (although partial results exists as identified in the related work).

Furthermore, as it is implausible that all queries in practice fall into a tractable class, for practical scalable algorithms, different approaches from direct query evaluation needs to be developed; for instance, based on (approximate) knowledge compilation that range from classical approaches to tensor factorizations and neural-symbolic approaches.

Existing systems for probabilistic query evaluation in PDBs need to be extended towards ontology-mediated queries. For FO-rewritable languages, the probabilistic relational database management systems can be used directly. It is not very clear what method would be suitable for other languages. Moreover, as pointed before, in some cases, although the query itself is unsafe, the database can be in a certain syntactical shape, ensuring tractability. Therefore, ideally, a system needs to incorporate tools to detect such cases.

Notice also that there is still a lot of work to be done on the semantic level to *develop probabilistic data models* that meet the requirements of individual application domains while remaining scalable to large instances. For instance, a data model based on the random worlds semantics (Grove, J. Y. Halpern, and Koller 1994), or alike can be a baseline for reasoning with rules and probabilistic facts. However, these approaches seem to be rather under-explored despite the fact that other models based on maximum entropy such as MLNs are very popular.

We have identified several connections between the research on probabilistic databases and that of question answering, knowledge base completion, rule mining, et cetera. These links need to be explored further in order to exploit the full potential in large-scale knowledge bases.

Bibliography

Abiteboul, Serge, T.-H. Hubert Chan, Evgeny Kharlamov, Werner Nutt, and Pierre Senellart (2011). "Capturing continuous data and answering aggregate queries in probabilistic XML". In *ACM Transactions on Database Systems (TODS)* **36**(4):1–45 (cited on page 148)

Abiteboul, Serge, Richard Hull, and Victor Vianu, editors (1995). *Foundations of Databases: The Logical Level*. 1st. Addison-Wesley Longman Publishing Co., Inc. (cited on page 33)

Amarilli, Antoine, Pierre Bourhis, and Pierre Senellart (2016). "Tractable Lineages on Treelike Instances: Limits and Extensions". In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-16)*. ACM, pages 355–370 (cited on page 50)

Amarilli, Antoine, Mikaël Monet, and Pierre Senellart (2017). "Conjunctive Queries on Probabilistic Graphs: Combined Complexity". In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-17)*, pages 217–232 (cited on page 45)

Arenas, Marcelo, Leopoldo E. Bertossi, and Jan Chomicki (1999). "Consistent Query Answers in Inconsistent Databases". In *Proceedings of the 18th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-99)*. ACM, pages 68–79 (cited on page 155)

Arora, Sanjeev and Boaz Barak (2009). *Computational Complexity: A Modern Approach*. 1st. Cambridge University Press (cited on page 21)

Artale, Alessandro, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev (2009). "The DL-Lite Family and Relations". In *Journal of Artificial Intelligence Research* **36**:1–69 (cited on pages 110, 113, 117)

Artale, Alessandro and Enrico Franconi (2005). "Temporal Description Logics". In *Handbook of Time and Temporal Reasoning in Artificial Intelligence*. Volume 1 of *Foundations of Artificial Intelligence*. Elsevier Science Publishers Ltd., pages 375–388 (cited on page 170)

Artale, Alessandro, Roman Kontchakov, Alisa Kovtunova, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev (2015). "First-Order Rewritability of Temporal Ontology-Mediated Queries". In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15)*. AAAI Press, pages 2706–2712 (cited on page 170)

Artale, Alessandro, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev (2015). "Tractable Interval Temporal Propositional and Description Logics". In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*. AAAI Press, pages 1417–1423 (cited on page 170)

Artale, Alessandro, Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev (2013). "Temporal Description Logic for Ontology-Based Data Access". In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-13)*. AAAI Press, pages 711–717 (cited on page 170)

Ashburner, Michael, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock (2000). "Gene Ontology: tool for the unification of biology". In *Nat Genet* **25**(1):25–29 (cited on page 113)

Baader, Franz (2014). "Ontology-Based Monitoring of Dynamic Systems". In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR-14)*. AAAI Press, pages 678–681 (cited on page 170)

Baader, Franz, Stefan Borgwardt, and Marcel Lippmann (2013). "Temporalizing Ontology-Based Data Access". In *Proceedings of the 24th International Conference on Automated Deduction (CADE-24)*. Volume 7898 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 330–344 (cited on page 170)

– (2015). "Temporal Query Entailment in the Description Logic SHQ". In *Journal of Web Semantics* **33**:71–93 (cited on page 175)

Baader, Franz, Sebastian Brandt, and Carsten Lutz (2005). "Pushing the EL Envelope". In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*. Professional Book Center, pages 364–369 (cited on pages 110, 113, 118)

Baader, Franz, Diego Calvanese, Deborah L McGuinness, Daniele Nardi, and Peter F Patel-Schneider, editors (2007). *The Description Logic Handbook: Theory, Implementation, and Applications*. 2nd. Cambridge University Press (cited on pages 9, 108, 110, 112, 117)

Baader, Franz, Silvio Ghilardi, and Carsten Lutz (2012). "LTL over Description Logic Axioms". In *ACM Transactions on Computational Logic* **13**(3):21:1–21:32 (cited on page 170)

Baader, Franz, Patrick Koopmann, and Anni-Yasmin Turhan (2017). "Using Ontologies to Query Probabilistic Numerical Data". In *Proceedings of 11th International Symposium on Frontiers of Combining Systems (FRoCos-17)*. Volume 10483 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 77–94 (cited on page 178)

Bacchus, Fahiem (1990). *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach to Probabilities*. MIT Press (cited on pages 4, 5)

Baget, Jean-Francois, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo (2011). "Walking the Complexity Lines for Generalized Guarded Existential Rules". In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 712–717 (cited on pages 109, 110, 117)

Bailey, Delbert D., Víctor Dalmau, and Phokion G. Kolaitis (2007). "Phase transitions of PP-complete satisfiability problems". In *Discrete Applied Mathematics* **155**(12): 1627–1639 (cited on page 25)

Beame, Paul, Jerry Li, Sudeepa Roy, and Dan Suciu (2017). "Exact Model Counting of Query Expressions: Limitations of Propositional Methods". In. Volume 42 of, 1:1–1:46 (cited on page 51)

Beame, Paul, Guy Van den Broeck, Dan Suciu, and Eric Gribkoff (2015). "Symmetric Weighted First-Order Model Counting". In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-15)*. ACM, pages 313–328 (cited on page 49)

Beeri, Catriel and Moshe Y. Vardi (1981). "The Implication Problem for Data Dependencies". In *Proceedings of the 8th International Colloquium on Automata, Languages, and Programming (ICALP-81)*. Volume 115 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 73–85 (cited on page 109)

Beigel, R., N. Reingold, and D. Spielman (1995). "PP Is Closed under Intersection". In *Journal of Computer and System Sciences* **50**(2):191–202 (cited on pages 25, 45, 142, 143, 165)

Bienvenu, Meghyn, Balder Ten Cate, Carsten Lutz, and Frank Wolter (2014). "Ontology-Based Data Access: A Study Through Disjunctive Datalog, CSP, and MMSNP". In *ACM Transactions on Database Systems (TODS)* **39**(4):33:1–33:44 (cited on pages 107, 114)

Bienvenu, Meghyn and Magdalena Ortiz (2015). "Ontology-Mediated Query Answering with Data-Tractable Description Logics". In *Reasoning Web. Web Logic Rules - 11th International Summer School*. Volume 9203 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 218–307 (cited on page 117)

Bishop, Christopher M (2006). *Pattern recognition and machine learning*. Springer-Verlag (cited on page 53)

Boole, George (1854). *An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*. Walton and Maberly (cited on page 3)

Bordes, Antoine, Jason Weston, Ronan Collobert, and Yoshua Bengio (2011). "Learning Structured Embeddings of Knowledge Bases." In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-11)*. Edited by Wolfram Burgard and Dan Roth. AAAI Press (cited on page 6)

Börger, Egon, Erich Grädel, and Yuri Gurevich (1997). *The Classical Decision Problem*. Springer-Verlag (cited on page 23)

Borgida, Alexander (1996). "On the Relative Expressiveness of Description Logics and Predicate Logics". In *Artificial Intelligence* **82**(1-2):353–367 (cited on page 112)

Borgwardt, Stefan, İsmail İlkan Ceylan, and Thomas Lukasiewicz (2017). "Ontology-Mediated Queries for Probabilistic Databases". In *Proceedings of the 31th AAAI Conference on Artificial Intelligence (AAAI-17)*. AAAI Press, pages 1063–1069 (cited on pages 10, 11)

Borgwardt, Stefan, Marcel Lippmann, and Veronika Thost (2013). "Temporal Query Answering in the Description Logic *DL-Lite*". In *Proceedings of the 9th International Symposium on Frontiers of Combining Systems (FroCoS 2013)*. Volume 8152 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 165–180 (cited on page 170)

Boulos, Jihad, Nilesh Dalvi, Bhushan Mandhani, Shobhit Mathur, Chris Ré, and Dan Suciu (2005). "MYSTIQ: A system for finding more answers by using probabilities". In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*. ACM, pages 891–893 (cited on page 51)

Cadoli, Marco and Francesco Donini (1997). "A survey on knowledge compilation". In *AI Communications* **10**(3-4):137–150 (cited on page 51)

Calì, Andrea, Georg Gottlob, and Michael Kifer (2013). "Taming the Infinite Chase: Query Answering under Expressive Relational Constraints". In *Journal of Artificial Intelligence Research* **48**:115–174 (cited on pages 9, 109, 110, 116, 119)

Calì, Andrea, Georg Gottlob, and Thomas Lukasiewicz (2012). "A General Datalog-Based Framework for Tractable Query Answering over Ontologies". In *Journal of Web Semantics* **14**:57–83 (cited on pages 9, 108, 109, 116, 119)

Calì, Andrea, Georg Gottlob, and Andreas Pieris (2012). "Towards More Expressive Ontology Languages: The Query Answering Problem". In *Artificial Intelligence* **193**: 87–128 (cited on pages 9, 109, 110, 116, 117, 119)

Calvanese, Diego, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo (2011). "The MASTRO System for Ontology-based Data Access". In *Semantic Web* **2**(1):43–53 (cited on page 113)

Calvanese, Diego, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati (2007). "Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family". In *Journal of Automated Reasoning* **39**(3): 385–429 (cited on page 117)

– (2013). "Data complexity of query answering in description logics". In *Artificial Intelligence* **195**:335–360 (cited on page 118)

Cantor, Georg (1891). "Ueber eine elementare Frage der Mannigfaltigkeitslehre". In *Jahresbericht der Deutschen Mathematiker-Vereinigung*: (cited on page 23)

Carnap, Rudolf (1950). *Logical Foundations of Probability*. University of Chicago Press (cited on page 3)

Ceylan, İsmail İlkan, Stefan Borgwardt, and Thomas Lukasiewicz (2017). "Most Probable Explanations for Probabilistic Database Queries". In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17)*. AAAI Press, pages 950–956 (cited on page 11)

Ceylan, İsmail İlkan, Adnan Darwiche, and Guy Van den Broeck (2016). "Open-World Probabilistic Databases". In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR-16)*. AAAI Press, pages 339–348 (cited on pages 10–12)

– (2017). "Open-World Probabilistic Databases: An Abridged Report". In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17)*. AAAI Press, pages 4796–4800 (cited on page 11)

Ceylan, İsmail İlkan, Thomas Lukasiewicz, and Rafael Peñaloza (2016). "Complexity Results for Probabilistic Datalog±". In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI-16)*. Volume 285 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, pages 1414–1422 (cited on pages 11, 155)

Ceylan, İsmail İlkan, Thomas Lukasiewicz, Rafael Peñaloza, and Oana Tifrea-Marciuska (2017). "Query Answering in Ontologies under Preference Rankings". In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17)*. AAAI Press, pages 943–949 (cited on page 12)

Ceylan, İsmail İlkan, Julian Mendez, and Rafael Peñaloza (2015). "The Bayesian Ontology Reasoner is BORN!" In *Proceedings of the 4th International Workshop on OWL Reasoner Evaluation (ORE-15)*. Volume 1387 of. CEUR-WS, pages 8–14 (cited on pages 12, 162)

Ceylan, İsmail İlkan and Rafael Peñaloza (2014a). "Bayesian Description Logics". In *Description Logics*. Volume 1193 of *CEUR Workshop Proceedings*. CEUR-WS, pages 447–458 (cited on page 12)

– (2014b). "The Bayesian Description Logic $\mathcal{BEL}$". In *Proceedings of the 7th International Joint Conference on Automated Reasoning (IJCAR-14)*. Volume 8562 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 480–494 (cited on pages 12, 151, 153, 166)

– (2014c). "Tight Complexity Bounds for Reasoning in the Description Logic $\mathcal{BEL}$". In *Proceedings of the 14th European Conference on Logics in Artificial Intelligence (JELIA-14)*. Volume 8761 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 77–91 (cited on page 12)

– (2015a). "Dynamic Bayesian Ontology Languages". In *CoRR* **abs/1506.08030**: (cited on page 12)

– (2015b). "Probabilistic Query Answering in the Bayesian Description Logic BEL". In *Proceedings of the 9th International Conference on Scalable Uncertainty Management (SUM-15)*. Volume 9310 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 21–35 (cited on page 12)

– (2017). "The Bayesian Ontology Language $\mathcal{BEL}$". In *Journal of Automated Reasoning* **58**(1):67–95 (cited on pages 12, 151, 156)

Chakraborty, Supratik, Dror Fried, Kuldeep S. Meel, and Moshe Y. Vardi (2015). "From weighted to unweighted model counting". In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15)*. AAAI Press, pages 689–695 (cited on page 50)

Chakraborty, Supratik, Kuldeep S. Meel, and Moshe Y. Vardi (2016). "Algorithmic Improvements in Approximate Counting for Probabilistic Inference: From Linear to Logarithmic SAT Calls". In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16)* (cited on page 149)

Chandra, Ashok K. and Philip M. Merlin (1977). "Optimal Implementation of Conjunctive Queries in Relational Data Bases". In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing (STOC-77)*. ACM, pages 77–90 (cited on pages 37, 116)

Chavira, Mark and Adnan Darwiche (2008). "On Probabilistic Inference by Weighted Model Counting". In *Artificial Intelligence* **172**(6-7):772–799 (cited on pages 4, 50)

Choi, Arthur, Adnan Darwiche, L Zheng, and Ole J Mengshoel (2011). "A tutorial on Bayesian networks for system health management". In *Machine Learning and Knowledge Discovery for Engineering Systems Health Management* **10**(1):1–29 (cited on page 171)

Church, Alonzo (1936). "An Unsolvable Problem of Elementary Number Theory". In *American Journal of Mathematics* **58**(2):345–363 (cited on page 23)

Codd, Edgar F (1972). *Relational completeness of data base sublanguages*. IBM Corporation (cited on page 33)

Cook, Stephen A. (1971). "The Complexity of Theorem-proving Procedures". In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC-71)*. ACM, pages 151–158 (cited on pages 24, 27)

– (1972). "A Hierarchy for Nondeterministic Time Complexity". In *Proceedings of the 4th Annual ACM Symposium on Theory of Computing (STOC-72)*. ACM, pages 187–192 (cited on page 25)

Costa, Paulo Cesar G. da, Kathryn B. Laskey, and Kenneth J. Laskey (2008). "PR-OWL: A Bayesian Ontology Language for the Semantic Web". In *Uncertainty Reasoning for the Semantic Web I, URSW 2005-2007*. Volume 5327 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 88–107 (cited on page 178)

Cozman, Fabio Gagliardi (2000). "Credal networks". In *Artificial Intelligence* **120**(2): 199–233 (cited on pages 59, 80, 154)

d'Amato, Claudia, Nicola Fanizzi, and Thomas Lukasiewicz (2008). "Tractable Reasoning with Bayesian Description Logics". In *Proceedings of the 2nd International Conference on Scalable Uncertainty Management (SUM-08)*. Springer-Verlag, pages 146–159 (cited on page 178)

Dagum, Paul, Adam Galper, and Eric Horvitz (1992). "Dynamic Network Models for Forecasting". In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-92)*. Morgan Kaufmann, pages 41–48 (cited on page 171)

Dalvi, Nilesh and Dan Suciu (2007). "Efficient Query Evaluation on Probabilistic Databases". In *The VLDB Journal* **16**(4):523–544 (cited on pages 41, 42, 50)
– (2012). "The dichotomy of probabilistic inference for unions of conjunctive queries". In *Journal of ACM* **59**(6):1–87 (cited on pages 40, 41, 43, 48, 66, 68–70, 79, 122, 146, 147, 184)

Darwiche, Adnan (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press (cited on pages 40, 50, 80, 86, 88, 104)

Darwiche, Adnan and Pierre Marquis (2002). "A Knowledge Compilation Map". In *Journal of Artificial Intelligence Research* **17**(1):229–264 (cited on page 51)

De Campos, Cassio Polpo and Fabio Gagliardi Cozman (2005). "The inferential complexity of bayesian and credal networks". In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1313–1318 (cited on page 80)

De Giacomo, Giuseppe and Moshe Y. Vardi (2013). "Linear Temporal Logic and Linear Dynamic Logic on Finite Traces". In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-13)*. AAAI Press, pages 854–860 (cited on page 175)

De Raedt, Luc, Anton Dries, Ingo Thon, Guy Van den Broeck, and Mathias Verbeke (2015). "Inducing Probabilistic Relational Rules from Probabilistic Examples". In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15)*. AAAI Press (cited on pages 6, 56)

De Raedt, Luc, Angelika Kimmig, and Hannu Toivonen (2007). "ProbLog: A probabilistic prolog and its application in link discovery". In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*. Morgan Kaufmann, pages 2468–2473 (cited on pages 9, 38, 50, 148)

De Virgilio, Roberto, Giorgio Orsi, Letizia Tanca, and Riccardo Torlone (2011). "Semantic Data Markets: A Flexible Environment for Knowledge Management". In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. ACM, pages 1559–1564 (cited on page 110)

Dean, Thomas and Keiji Kanazawa (1989). "A Model for Reasoning About Persistence and Causation". In *Computational Intelligence* **5**(3):142–150 (cited on pages 170, 171)

Degris, Thomas and Oliver Sigaud (2010). "Factored Markov Decision Processes". In *Markov Decision Processes in Artificial Intelligence*. Chapter 4, pages 99–126 (cited on page 179)

Dong, Xin, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang (2014). "Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion". In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 601–610 (cited on pages 5–8)

Dong, Xin, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang (2015). "Knowledge-based Trust: Estimating the Trustworthiness of Web Sources". In *Proceedings of VLDB Endowment* **8**(9):938–949 (cited on page 5)

Dong, Xin, Alon Y. Halevy, and Cong Yu (2007). "Data Integration with Uncertainty". In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB-07)*. VLDB Endowment, pages 687–698 (cited on page 37)

Durand, Arnaud, Miki Hermann, and Phokion G. Kolaitis (2005). "Subtractive reductions and complete problems for counting complexity classes". In *Theoretical Computer Science* **340**(3):496–513 (cited on page 28)

Durrett, Rick (2010). *Probability: Theory and examples*. 4th Edition. Cambridge University Press (cited on page 18)

Ebbinghaus, H-D., Jörg Flum, and Wolfgang Thomas (1994). *Mathematical Logic*. 2nd ed. Springer-Verlag (cited on page 13)

Eiter, Thomas and Georg Gottlob (1995). "The Complexity of Logic-based Abduction". In *Journal of ACM* **42**(1):3–42 (cited on pages 103, 104)

Eiter, Thomas, Georg Gottlob, Magdalena Ortiz, and Mantas Šimkus (2008). "Query Answering in the Description Logic Horn-SHIQ". In *Proceedings of the 11th European Conference on Logics in Artificial Intelligence (JELIA-08)*. Springer-Verlag, pages 166–179 (cited on page 118)

Eiter, Thomas, Thomas Lukasiewicz, and Livia Predoiu (2016). "Generalized Consistent Query Answering under Existential Rules". In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR-16)*. AAAI Press (cited on page 139)

Enderton, Herbert (2001). *A Mathematical Introduction to Logic*. 2nd ed. Harcourt/Academic Press (cited on page 13)

Fader, Anthony, Stephen Soderland, and Oren Etzioni (2011). "Identifying Relations for Open Information Extraction". In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1535–1545 (cited on page 5)

Fagin, Ronald, Joseph Y. Halpern, and Nimrod Megiddo (1990). "A logic for reasoning about probabilities". In *Information and Computation* **87**(1):78–128. Special Issue: Selections from 1988 IEEE Symposium on Logic in Computer Science (cited on page 4)

Fagin, Ronald, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa (2005). "Data Exchange: Semantics and Query Answering". In *Theoretical Computer Science* **336**(1): 89–124 (cited on pages 109, 110, 116, 119)

Ferrucci, D. A. (2012). "Introduction to "This is Watson"". In *IBM J. Res. Dev.* **56**(3): 235–249 (cited on page 5)

Ferrucci, David, Anthony Levas, Sugato Bagchi, David Gondek, and Erik T. Mueller (2013). "Watson: Beyond jeopardy!" In *Artificial Intelligence* **199-200**:93–105 (cited on page 5)

Fink, Robert, Andrew Hogue, Dan Olteanu, and Swaroop Rath (2011). "SPROUT2: A Squared Query Engine for Uncertain Web Data". In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. ACM, pages 1299–1302 (cited on page 51)

Fink, Robert and Dan Olteanu (2016). "Dichotomies for Queries with Negation in Probabilistic Databases". In *ACM Transactions on Database Systems (TODS)* **41**(1): 4:1–4:47 (cited on pages 50, 70)

Fink, Robert, Dan Olteanu, and S. Rath (2011). "Providing support for full relational algebra in probabilistic databases". In *Proceedings of the 27th International Conference on Data Engineering (ICDE-11)*, pages 315–326 (cited on page 50)

Fitting, Melvin (1996). *First-order Logic and Automated Theorem Proving*. Springer-Verlag (cited on pages 13, 16)

Fremont, Daniel, Markus Rabe, and Sanjit Seshia (2017). "Maximum Model Counting". In *Proceedings of the 31th AAAI Conference on Artificial Intelligence (AAAI-17)* (cited on page 149)

Frisch, Alan M. and Peter Haddawy (1994). "Anytime Deduction for Probabilistic Logic". In *Artificial Intelligence* **69**(1-2):93–122 (cited on page 4)

Fuhr, Norbert and T Rölleke (1997). "A probabilistic relational algebra for the integration of information retrieval and database systems". In *ACM Transactions on Database Systems (TOIS)* **15**(1):32–66 (cited on page 50)

Fürer, Martin (1983). "The computational complexity of the unconstrained limited domino problem (with implications for logical decision problems)". In *Logic and Machines* (cited on page 139)

Furst, Merrick, James B. Saxe, and Michael Sipser (1984). "Parity, circuits, and the polynomial-time hierarchy". In *Mathematical systems theory* **17**(1):13–27 (cited on page 30)

Galárraga, Luis, Simon Razniewski, Antoine Amarilli, and Fabian M. Suchanek (2017). "Predicting Completeness in Knowledge Bases". In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining (WSDM-17)*. ACM, pages 375–383 (cited on page 7)

Galárraga, Luis, Christina Teflioudi, Katja Hose, and Fabian Suchanek (2013). "Amie: association rule mining under incomplete evidence in ontological knowledge bases". In *Proceedings of the 22nd international conference on World Wide Web (WWW-13)*, pages 413–422 (cited on page 56)

Gentzen, Gerhard (1935). "Untersuchungen über das logische Schließen. I". In *Mathematische Zeitschrift* **39**(1):176–210 (cited on page 16)

Getoor, Lise and Ben Taskar (2007). *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press (cited on page 4)

Gill, John T. (1977). "Computatonal Complexity of Probabilistic Turing Machines". In *SIAM Journal on Computing* **6**(4):675–695 (cited on page 25)

Glimm, Birte, Ian Horrocks, Carsten Lutz, and Ulrike Sattler (2008). "Conjunctive Query Answering for the Description Logic SHIQ". In *Journal of Artificial Intelligence Research* **31**:157–204 (cited on page 118)

Glimm, Birte, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang (2014). "HermiT: An OWL 2 Reasoner". In *Journal of Automated Reasoning* **53**(3): (cited on page 113)

Glimm, Birte, Yevgeny Kazakov, and Carsten Lutz (2011). "Status QIO: An Update". In *Proceedings of the 24th International Workshop on Description Logics (DL-11)*. Volume 745 of. CEUR-WS (cited on page 118)

Gödel, Kurt (1929). "Über die Vollständigkeit des Logikkalküls". Doctoral dissertation. University Of Vienna (cited on page 16)

Gomes, Carla P, Ashish Sabharwal, and Bart Selman (2006). "Model counting: A new strategy for obtaining good bounds". In *Proceedings of The 21st National Conference on Artificial Intelligence(AAAI-06)*. AAAI Press, pages 54–61 (cited on page 51)

Gottlob, Georg, Nicola Leone, and Francesco Scarcello (2001). "The Complexity of Acyclic Conjunctive Queries". In *Journal of ACM* **48**(3):431–498 (cited on page 37)

Gottlob, Georg, Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari (2013). "Query answering under probabilistic uncertainty in Datalog± ontologies". In *Annals of Mathematics and Artificial Intelligence* **69**(1):37–72 (cited on page 148)

Gottlob, Georg, Marco Manna, and Andreas Pieris (2013). "Combining Decidability Paradigms for Existential Rules". In *In Proceedings of the 29thth International Conference on Logic Programming (ICLP-13)*. Cambridge University Press, pages 877–892 (cited on pages 110, 117)

– (2014). "Polynomial Combined Rewritings for Existential Rules". In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR-14)*. AAAI Press, pages 268–277 (cited on pages 110, 117)

Gottlob, Georg, Sebastian Rudolph, and Mantas Simkus (2014). "Expressiveness of Guarded Existential Rule Languages". In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-14)*. ACM, pages 27–38 (cited on pages 110, 117)

Grädel, Erich, Yuri Gurevich, and Colin Hirsch (1998). "The complexity of query reliability". In *Proceedings of the 17th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-98)*. ACM, pages 227–234 (cited on pages 41, 50)

Grädel, Erich, Phokion G. Kolaitis, and Moshe Y. Vardi (1997). "On the Decision Problem for Two-Variable First-Order Logic". In *The Bulletin of Symbolic Logic* **3**(1): 53–69 (cited on page 25)

Greenemeier, Larry (2015). “Human Traffickers Caught on Hidden Internet”. In *Scientific American*: (cited on page 5)

Gribkoff, Eric and Dan Suciu (2016a). “SlimShot: In-Database Probabilistic Inference for Knowledge Bases”. In *Proceedings of VLDB Endowment* **9**(7): (cited on pages 9, 119)
– (2016b). “SlimShot: In-Database Probabilistic Inference for Knowledge Bases”. In *Proceedings of VLDB Endowment* **9**(7): (cited on page 52)

Gribkoff, Eric, Dan Suciu, and Guy Van den Broeck (2014). “Lifted Probabilistic Inference : A Guide for the Database Researcher”. In *IEEE Data Engineering Bulletin* **37**:6–17 (cited on page 50)

Gribkoff, Eric, Guy Van den Broeck, and Dan Suciu (2014a). “The most probable database problem”. In *Proceedings of the 1st International Workshop on Big Uncertain Data (BUDA)* (cited on pages 87, 90, 93, 94, 104, 149)
– (2014b). “Understanding the Complexity of Lifted Inference and Asymmetric Weighted Model Counting”. In *Proceedings of the 30th Annual Conference on Uncertainty in Artificial Intelligence (UAI-14)*. AUAI Press, pages 280–289 (cited on pages 43, 51, 66, 68, 73)

Group, W3C OWL Working (2012). *OWL 2 Web Ontology Language. Document overview (Second Edition)* (cited on page 113).
URL: `http://www.w3.org/TR/owl2-overview/`

Grove, Adam J., Joseph Y. Halpern, and Daphne Koller (1994). “Random Worlds and Maximum Entropy”. In *Journal of Artificial Intelligence Research* **2**(1):33–88 (cited on pages 148, 185)

Hailperin, Theodore (July 1984). “Probability logic”. In *Notre Dame Journal of Formal Logic* **25**(3):198–212 (cited on page 3)

Halpern, Joseph Y. (1990). “An analysis of first-order logics of probability”. In *Artificial Intelligence* **46**(3):311–350 (cited on page 4)
– (2003). *Reasoning about uncertainty*. MIT Press (cited on pages 4, 5, 59)

Halpern, Joseph Y., Robert Harper, Neil Immerman, Phokion G. Kolaitis, Moshe Y. Vardi, and Victor Vianu (June 2001). “On the Unusual Effectiveness of Logic in Computer Science”. In *Bulletin of Symbolic Logic* **7**(2):213–236 (cited on page 3)

Hemachandra, Lane A. (1989). “The Strong Exponential Hierarchy Collapses”. In *Journal of Computer and System Sciences* **39**(3):299–322 (cited on pages 139, 140)

Hesse, William, Eric Allender, and David A. Mix Barrington (2002). “Uniform constant-depth threshold circuits for division and iterated multiplication”. In *Journal of Computer and System Sciences* **65**(4):695–716 (cited on pages 30, 90, 94)

Hilbert, David (1923). “Die logischen Grundlagen der Mathematik.” In *Mathematische Annalen* **88**:151–165 (cited on page 16)

Hinrichs, Timothy and Michael Genesereth (2006). *Herbrand Logic*. Technical report LG-2006-02. Stanford University (cited on page 34)

Hintikka, Jaakko (1969). "Semantics for Propositional Attitudes". In. *Philosophical Logic*. Springer-Verlag, pages 21–45 (cited on page 50)

Hoffart, Johannes, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum (2013). "YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia". In *Artificial Intelligence* **194**:28–61 (cited on page 5)

Hopcroft, J. E. and J. D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company (cited on page 21)

Horrocks, Ian, Oliver Kutz, and Ulrike Sattler (2006). "The Even More Irresistible SROIQ". In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-06)*. AAAI Press, pages 57–67 (cited on page 110)

Horrocks, Ian, Peter F. Patel-Schneider, and Frank Van Harmelen (2003). "From SHIQ and RDF to OWL: The Making of a Web Ontology Language". In *Journal of Web Semantics* **1**:2003 (cited on page 113)

Husmeier, Dirk (2003). "Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks". In *Bioinformatics* **19**(17):2271–2282 (cited on page 171)

Imieliński, Tomasz and Witold Lipski (1984). "Incomplete Information in Relational Databases". In *Journal of ACM* **31**(4):761–791 (cited on pages 50, 147)

Immerman, Neil (1986). "Relational queries computable in polynomial time". In *Information and Control* **68**(1):86–104 (cited on page 116)

– (1999). *Descriptive complexity*. Springer-Verlag (cited on pages 30, 36, 94)

Jaeger, Manfred (1997). "Relational Bayesian Networks". In *Proceedings of the 23rd Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*. Morgan Kaufmann, pages 266–273 (cited on pages 9, 178)

Jha, Abhay and Dan Suciu (2013). "Knowledge Compilation Meets Database Theory: Compiling Queries to Decision Diagrams". In *Theory of Computing Systems* **52**(3): (cited on page 51)

Jung, Jean Christoph and Carsten Lutz (2012). "Ontology-based Access to Probabilistic Data with OWL QL". In *Proceedings of the 11th International Conference on The Semantic Web - Volume Part I*. Springer-Verlag, pages 182–197 (cited on pages 125, 148, 178)

Karp, Richard M. (1973). "Reducibility among combinatorial problems". In *Complexity of Computer Computations*:219–241 (cited on page 24)

Kazakov, Yevgeny, Markus Krötzsch, and František Simančík (2012). "ELK Reasoner: Architecture and Evaluation". In *Proceedings of the OWL Reasoner Evaluation Workshop 2012*. CEUR-WS (cited on page 113)

Kimelfeld, Benny and Pierre Senellart (2013). "Probabilistic XML : Models and Complexity". In:39–66 (cited on page 148)

Koller, Daphne and N. Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press (cited on pages 50, 83, 84, 95, 149, 165)

Koller, Daphne, Alon Y. Levy, and Avi Pfeffer (1997). "P-CLASSIC: A Tractable Probablistic Description Logic." In *Proceedings of The 14th National Conference on Artificial Intelligence(AAAI-97)*. AAAI Press, pages 390–397 (cited on page 178)

Kolmogorov, A. N. (1933). *Grundbegriffe der Wahrscheinlichkeitsrechnung*. Springer-Verlag (cited on pages 3, 17)

Kripke, Saul A. (1963). "Semantical Analysis of Modal Logic I. Normal Propositional Calculi". In *Zeitschrift fur mathematische Logik und Grundlagen der Mathematik* **9**(56): 67–96 (cited on page 50)

Krisnadhi, Adila and Carsten Lutz (2007). "Data Complexity in the $\mathcal{EL}$ Family of Description Logics". In *Proceedings of 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-07)*. Volume 4790 of. Springer-Verlag, pages 333–347 (cited on page 117)

Krompaß, Denis, Maximilian Nickel, and Volker Tresp (2014). "Querying Factorized Probabilistic Triple Databases". In *Proceedings of the 13th International Semantic Web Conference (ISWC-14)*. Springer-Verlag, pages 114–129 (cited on page 51)

Krötzsch, Markus and Sebastian Rudolph (2007). "Conjunctive queries for EL with role composition". In *Proceedings of the 20th International Workshop on Description Logics (DL-07)*. CEUR-WS (cited on page 117)

– (2011). "Extending Decidable Existential Rules by Joining Acyclicity and Guardedness". In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*. AAAI Press, pages 963–968 (cited on pages 109, 110, 117)

Ku, Joy P., Jennifer L. Hicks, Trevor Hastie, Jure Leskovec, Christopher Ré, and Scott L. Delp (2015). "The mobilize center: An NIH big data to knowledge center to advance human movement research and improve mobility". In *Journal of the American Medical Informatics Association* **22**(6):1120–1125 (cited on page 5)

Kwisthout, Johan (2011). "Most probable explanations in Bayesian networks: Complexity and tractability". In *International Journal of Approximate Reasoning* **52**(9):1452–1469 (cited on page 104)

Ladner, R.E., N.A. Lynch, and A.L. Selman (1975). "A comparison of polynomial time reducibilities". In *Theoretical Computer Science* **1**:103–123 (cited on page 28)

Langmead, Christopher James. (2009). "Generalized queries and Bayesian statistical model checking in dynamic Bayesian networks: Application to personalized medicine". In *Proceedings of 9th Annual International Conference on Computational Systems Bioinformatics (CSB)*, pages 201–212 (cited on pages 174, 175)

Lembo, Domenico, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo (2010). "Inconsistency-Tolerant Semantics for Description Logics". In *Proceedings of the 4th International Conference on Web Reasoning and Rule Systems (RR-10)*. Volume 6333 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 103–117 (cited on page 155)

Lerner, Uri, Ronald Parr, Daphne Koller, and Gautam Biswas (2000). "Bayesian Fault Detection and Diagnosis in Dynamic Systems". In *Proceedings of The 17th National Conference on Artificial Intelligence(AAAI-00)*. AAAI Press, pages 531–537 (cited on page 171)

Levi, Isaac (1980). *The Enterprise of Knowledge*. MIT Press (cited on page 53)

Levin, Leonid A. (1973). "Universal search problems". In *Problems of Information Transmission* **9**(3): (cited on page 24)

Lewis, Harry R. (1980). "Complexity results for classes of quantificational formulas". In *Journal of Computer and System Sciences* **21**(3):317–353 (cited on page 25)

Libkin, Leonid (Aug. 2004). *Elements of Finite Model Theory*. Springer-Verlag (cited on pages 17, 33)

Littman, Michael L. (1999). "Initial experiments in stochastic satisfiability". In *Proceedings of The 16th National Conference on Artificial Intelligence(AAAI-99)* (cited on page 86)

Littman, Michael L., Thomas Dean, and Leslie Pack Kaelbling (1995). "On the Complexity of Solving Markov Decision Problems". In *Proceedings of the 21rd Annual Conference on Uncertainty in Artificial Intelligence (UAI-95)*. Morgan Kaufmann, pages 394–402 (cited on page 179)

Littman, Michael L., Judy Goldsmith, and Martin Mundhenk (1998). "The computational complexity of probabilistic planning". In *Journal of Artificial Intelligence Research* **9**: 1–36 (cited on page 27)

Littman, Michael L., Stephen M. Majercik, and Toniann Pitassi (2001). "Stochastic Boolean Satisfiability". In *Journal of Automated Reasoning* **27**(3):251–296 (cited on pages 25, 84)

Lukasiewicz, Thomas, Maria Vanina Martinez, Andreas Pieris, and Gerardo I. Simari (2015). "From Classical to Consistent Query Answering under Existential Rules". In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*. Edited by Blai Bonet and Sven Koenig. AAAI Press, pages 1546–1552 (cited on pages 116, 119)

Lukasiewicz, Thomas and Umberto Straccia (2008). "Managing Uncertainty and Vagueness in Description Logics for the Semantic Web". In *Journal of Web Semantics* **6**(4): 291–308 (cited on page 178)

Lutz, Carsten (2008). "The Complexity of Conjunctive Query Answering in Expressive Description Logics". In *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR-08)*. Springer-Verlag, pages 179–193 (cited on page 118)

Lutz, Carsten, Frank Wolter, and Michael Zakharyaschev (2008). "Temporal Description Logics: A Survey". In *Proceedings of the 15th International Symposium on Temporal Representation and Reasoning (TIME-08)*. IEEE Computer Society, pages 3–14 (cited on page 170)

Marsden, Rhodri (2015). "Can Google's The Knowledge Vault automate the process of determining the 'truth'?" In *Independent*: (cited on page 5)

Milch, Brian, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L Ong, and Andrey Kolobov (2007). "BLOG: Probabilistic Models with Unknown Objects". In *Statistical relational learning*:373 (cited on page 80)

Mintz, Mike, Steven Bills, Rion Snow, and Dan Jurafsky (2009). "Distant Supervision for Relation Extraction Without Labeled Data". In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, pages 1003–1011 (cited on page 6)

Mitchell, T., W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling (2015). "Never-Ending Learning". In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*, pages 2302–2310 (cited on page 5)

Mundhenk, Martin, Judy Goldsmith, Christopher Lusena, and Eric Allender (2000). "Complexity of Finite-horizon Markov Decision Process Problems". In *Journal of ACM* **47**(4):681–720 (cited on page 179)

Munroe, Randall (2015). *Google's Datacenters on Punch Cards* (cited on page 57)

Murphy, Kevin (2002). "Dynamic bayesian networks: representation, inference and learning". PhD thesis. University of California, Berkeley (cited on pages 170, 171, 173)

Nachimuthu, Senthil K. and Peter J. Haug (2012). "Early Detection of Sepsis in the Emergency Department using Dynamic Bayesian Networks". In *Proceedings of the Annual Symposium of American Medical Informatics Association (AMIA-12)*, pages 653–662 (cited on page 171)

Nenov, Yavor, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee (2015). "RDFox: A Highly-Scalable RDF Store". In *International Semantic Web Conference (2)*. Volume 9367 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 3–20 (cited on page 110)

Nilsson, Nils J. (1986). "Probabilistic Logic". In *Artificial Intelligence* **28**(1):71–88 (cited on page 4)

– (1998). *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann (cited on page 84)

Niu, Feng, Christopher Ré, AnHai Doan, and Jude W. Shavlik (2011). "Tuffy: Scaling up Statistical Inference in Markov Logic Networks using an RDBMS". In *Proceedings of VLDB Endowment* **4**(6):373–384 (cited on pages 51, 52)

Olteanu, Dan and Jiewen Huang (2008). "Using OBDDs for Efficient Query Evaluation on Probabilistic Databases". In *Proceedings of the 2nd International Conference on Scalable Uncertainty Management (SUM-08)*. Volume 5291 of *Lecture Notes in Computer Science*, pages 326–340 (cited on page 50)

– (2009). "Secondary-storage Confidence Computation for Conjunctive Queries with Inequalities". In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. ACM, pages 389–402 (cited on page 50)

Olteanu, Dan and Maximilian Schleich (2016). "Factorized Databases". In *SIGMOD Record* **45**(2):5–16 (cited on page 51).
DOI: 10.1145/3003665.3003667

Ortiz, Magdalena, Sebastian Rudolph, and Mantas Šimkus (2011). "Query Answering in the Horn Fragments of the Description Logics SHOIQ and SROIQ". In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*. AAAI Press, pages 1039–1044 (cited on page 118)

Papadimitriou, Christos H. (1994). *Computational complexity*. Addison-Wesley (cited on page 21)

Park, James D. and Adnan Darwiche (2004a). "A differential semantics for jointree algorithms". In *AIJ* **156**(2):197–216 (cited on page 149)

– (2004b). "Complexity Results and Approximation Strategies for MAP Explanations". In *Journal of Artificial Intelligence Research* **21**(1):101–133 (cited on pages 27, 86, 165, 173)

Pearl, Judea (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann (cited on pages 3, 50, 83, 84, 149, 165)

Peters, Shanan E., Ce Zhang, Miron Livny, and Christopher Ré (2014). "A Machine Reading System for Assembling Synthetic Paleontological Databases." In *PLoS ONE* **9**(12): (cited on page 5)

Pipatsrisawat, Knot and Adnan Darwiche (2009). "A New d-DNNF-based Bound Computation Algorithm for Functional E-MAJSAT". In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*. Morgan Kaufmann, pages 590–595 (cited on page 149)

Pisanelli, Domenico M (2004). *Ontologies in medicine*. Volume 102. IOS Press (cited on page 152)

Pnueli, Amir (1977). "The Temporal Logic of Programs". In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, pages 46–57 (cited on pages 171, 174, 175)

Poggi, Antonella, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and (2008). "Linking Data to Ontologies". In *Journal on Data Semantics* **10**: (cited on pages 9, 107, 147)

Poole, David (1997). "The independent choice logic for modelling multiple agents under uncertainty". In *Artificial Intelligence* **94**(1-2):7–56 (cited on pages 38, 50, 119, 147)

– (2003). “First-order probabilistic inference”. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*. Volume 3 of, pages 985–991 (cited on page 81)

Poon, H and Pedro Domingos (2006). “Sound and Efficient Inference with Probabilistic and Deterministic Dependencies”. In *Proceedings of The 21st National Conference on Artificial Intelligence(AAAI-06)*. AAAI Press, pages 458–463 (cited on page 52)

Post, Emil L. (1944). “Recursively enumerable sets of positive integers and their decision problems”. In *Bulletin of the American Mathematical Society* **50**(5):284–317 (cited on page 24)

Provan, J. Scott and Michael O. Ball (1983). “The Complexity of Counting Cuts And Of Computing The Probability That A Graph Is Connected”. In *SIAM Journal on Computing* **12**(4):777–788 (cited on page 28)

Raedt, Luc De, Kristian Kersting, and Sriraam Natarajan (2016). *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*. Morgan & Claypool Publishers (cited on page 4)

Ré, Christopher and Dan Suciu (2009). “The trichotomy of HAVING queries on a probabilistic database”. In *The VLDB Journal* **18**(5):1091–1116 (cited on page 50)

Reiter, Raymond (1978). “On closed world data bases”. In *Logic and Data Bases*:55–76 (cited on pages 7, 34, 53, 59)

– (1980). “A logic for default reasoning”. In *Artificial Intelligence* **13**(1):81–132 (cited on page 59)

Richardson, Matthew and Pedro Domingos (2006). “Markov Logic Networks”. In *Machine Learning* **62**(1):107–136 (cited on pages 9, 104, 148, 178)

Riguzzi, Fabrizio, Elena Bellodi, Evelina Lamma, and Riccardo Zese (2015). “Probabilistic Description Logics under the Distribution Semantics”. In *Semantic Web Journal* **6**(5): 477–501 (cited on page 178)

Robinson, J. A. (1965). “A Machine-Oriented Logic Based on the Resolution Principle”. In *Journal of ACM* **12**(1):23–41 (cited on page 16)

Rosati, Ricardo (2007). “On conjunctive query answering in EL”. In *Proceedings of the 20th International Workshop on Description Logics (DL-07)*. CEUR-WS (cited on page 117)

Rose, Cedric, Cherif Smaili, and Francois Charpillet (2005). “A Dynamic Bayesian Network for Handling Uncertainty in a Decision Support System Adapted to the Monitoring of Patients Treated by Hemodialysis”. In *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-05)*. IEEE Computer Society, pages 594–598 (cited on page 171)

Rossman, Benjamin (2008). “Homomorphism preservation theorems”. In *Journal of ACM* **55**(3):1–53 (cited on page 115)

Roth, Dan (1996). "On the Hardness of Approximate Reasoning". In *Artificial Intelligence* **82**(1-2):273–302 (cited on page 163)

Russell, Stuart (2015). "Unifying Logic and Probability". In *Communications of ACM* **58**(7):88–97 (cited on page 4)

Sagiv, Yehoshua and Mihalis Yannakakis (1980). "Equivalences Among Relational Expressions with the Union and Difference Operators". In *Journal of ACM* **27**(4): 633–655 (cited on page 68)

Sang, Tian, Paul Beame, and Henry Kautz (2007). "A Dynamic Approach to MPE and Weighted MAX-SAT". In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*. Morgan Kaufmann, pages 173–179 (cited on page 149)

Sato, Taisuke (1995). "A Statistical Learning Method for Logic Programs with Distribution Semantics". In *In Proceedings of the 12th International Conference on Logic Programming (ICLP-95)*. MIT Press, pages 715–729 (cited on pages 38, 50, 147)

Sato, Taisuke and Yoshitaka Kameya (1997). "PRISM: A Language for Symbolic-statistical Modeling". In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*. Morgan Kaufmann, pages 1330–1335 (cited on pages 38, 50)

Savitch, Walter J. (1970). "Relationships between nondeterministic and deterministic tape complexities". In *Journal of Computer and System Sciences* **4**(2):177–192 (cited on page 25)

Schild, Klaus (1991). "A correspondence theory for terminological logics: Preliminary report". In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*. Morgan Kaufmann, pages 466–471 (cited on page 118)

Schmidt-Schauß, Manfred and Gert Smolka (1991). "Attributive concept descriptions with complements". In *Artificial Intelligence* **48**(1):1–26 (cited on page 110)

Selman, Bart and Henry Kautz (1996). "Knowledge compilation and theory approximation". In *Journal of ACM* **43**(2):193–224 (cited on page 51)

Shapiro, Norman (1956). "Degrees of Computability". In *Transactions of the American Mathematical Society* **82**(1):281–299 (cited on page 24)

Shimony, Eyal Solomon (1994). "Finding MAPs for Belief Networks is NP-hard". In *Artificial Intelligence* **68**(2):399–410 (cited on pages 86, 167)

Shin, Jaeho, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré (2015). "Incremental Knowledge Base Construction Using DeepDive". In. Volume 8 of. VLDB Endowment, pages 1310–1321 (cited on pages 5, 7, 51, 52)

Simon, Janos (1977). "On the difference between one and many". In *Proceedings of the 4th International Colloquium on Automata, Languages, and Programming (ICALP-77)* (18):480–491 (cited on page 28)

Sipser, Michael (1996). *Introduction to the Theory of Computation*. 1st. International Thomson Publishing (cited on pages 21, 22)

Sirin, Evren, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz (2007). "Pellet: A Practical OWL-DL Reasoner". In *Journal of Web Semantics* **5**(2): 51–53 (cited on page 113)

Smullyan, Raymond M. (1968). *First-Order Logic*. Springer-Verlag (cited on page 16)

Socher, Richard, Danqi Chen, Christopher D Manning, and Andrew Ng (2013). "Reasoning With Neural Tensor Networks for Knowledge Base Completion". In *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., pages 926–934 (cited on pages 6, 51)

Stanisław, Jaśkowski (1934). "On the Rules of Suppositions in Formal Logic". In *Studia Logica* (1):5–32 (cited on page 16)

Staworko, Sławomir and Jan Chomicki (2010). "Consistent Query Answers in the Presence of Universal Constraints". In *Inf. Syst.* **35**(1):1–22 (cited on page 108)

Steigmiller, Andreas, Thorsten Liebig, and Birte Glimm (2014). "Konclude: System description". In *Web Semantics: Science, Services and Agents on the World Wide Web* **27**:78–85 (cited on page 113)

Stockmeyer, Larry J. (1976). "The polynomial-time hierarchy". In *Theoretical Computer Science* **3**(1):1–22 (cited on pages 26, 92)

Suciu, Dan, Dan Olteanu, Christopher Ré, and Christoph Koch (2011). *Probabilistic Databases*. Volume 3, pages 1–180 (cited on pages 7, 37, 38, 43, 50, 79, 142, 147)

Sutton, Charles and Andrew McCallum (2011). "An introduction to conditional random fields". In *Machine Learning* **4**(4):267–373 (cited on page 56)

Tobies, S. (2001). "Complexity Results and Practical Algorithms for Logics in Knowledge Representation". PhD thesis. Theoretical Computer Science, RWTH-Aachen (cited on page 113)

Toda, Seinosuke (1989). "On the computational power of PP and +P". In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 514–519 (cited on pages 28, 78, 79, 141)

Toda, Seinosuke and Osamu Watanabe (1992). "Polynomial-time 1-Turing reductions from #PH to #P". In *Theoretical Computer Science* **100**(1):205–221 (cited on page 28)

Torán, Jacobo (1991). "Complexity classes defined by counting quantifiers". In *Journal of ACM* **38**(3):753–774 (cited on page 27)

Trakhtenbrot, Boris A. (1950). " The impossibility of an algorithm for the decidability problem on finite classes". In *Doklady AN SSR* **70**(4):569–572 (cited on pages 23, 43)

Tsarkov, Dmitry and Ian Horrocks (2006). "FaCT++ Description Logic Reasoner: System Description". In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR-06)*. Springer-Verlag, pages 292–297 (cited on page 113)

Tseitin, G. S. (1968). "On the complexity of derivations in the propositional calculus". In *Studies in Mathematics and Mathematical Logic* **Part II**:115–125 (cited on page 72)

Turing, Alan M. (1936). "On Computable Numbers, with an Application to the Entscheidungsproblem". In *Proceedings of the London Mathematical Society* **2**(42):230–265 (cited on pages 22, 23)

Umans, Christopher (2001). "The Minimum Equivalent DNF Problem and Shortest Implicants". In *Journal of Computer and System Sciences* **63**(4):597–611 (cited on page 104)

Valiant, Leslie Gabriel (1979a). "The complexity of computing the permanent". In *Theoretical Computer Science* **8**(2):189–201 (cited on page 27)

– (1979b). "The Complexity of Enumeration And Reliability Problems". In *SIAM Journal on Computing* **8**(3):410–421 (cited on page 28)

Van Dalen, Dirk (2004). *Logic and structure*. 4th ed. Springer-Verlag (cited on page 13)

Van den Broeck, Guy, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt (2011). "Lifted Probabilistic Inference by First-Order Knowledge Compilation". In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 2178–2185 (cited on page 51)

Vardi, Moshe Y. (1982). "The Complexity of Relational Query Languages". In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC-82)*. Edited by Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber. ACM, pages 137–146 (cited on pages 36, 115, 116)

Vollmer, Heribert (1999). *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag (cited on page 28)

Wagner, Klaus W. (1986). "The complexity of combinatorial problems with succinct input representation". In *Acta Informatica* **23**(3):325–356 (cited on pages 27, 30, 44, 46, 75, 123, 130, 143)

Wang, William Yang, Kathryn Mazaitis, and William W. Cohen (2013). "Programming with Personalized Pagerank: A Locally Groundable First-order Probabilistic Logic". In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. ACM, pages 2129–2138 (cited on pages 7, 56)

Weikum, Gerhard, Johannes Hoffart, and Fabian Suchanek (2016). "Ten Years of Knowledge Harvesting: Lessons and Challenges". In *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* **39**(3):41–50 (cited on page 6)

Wu, Wentao, Hongsong Li, Haixun Wang, and Kenny Q. Zhu (2012). "Probase: A Probabilistic Taxonomy for Text Understanding". In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, pages 481–492 (cited on page 5)

Yannakakis, Mihalis (1981). "Algorithms for Acyclic Database Schemes". In *Proceedings of the 7th International Conference on Very Large Data Bases (VLDB-81)*. VLDB Endowment, pages 82–94 (cited on page 37)

Zhou, Yujiao, Yavor Nenov, Bernardo Cuenca Grau, and Ian Horrocks (2014). "Pay-As-You-Go OWL Query Answering Using a Triple Store". In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14)*. AAAI Press, pages 1142–1148 (cited on page 113)