10 Years of Probabilistic Querying – What Next?

Martin Theobald[1], Luc De Raedt[2], Maximilian Dylla[3], Angelika Kimmig[2], Iris Miliaraki[3]

[1] Universiteit Antwerpen, Middelheimlaan 1, 2020 Antwerp, Belgium
martin.theobald@ua.ac.be
[2] Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
{luc.deraedt,angelika.kimmig}@cs.kuleuven.be
[3] Max Planck Institut Informatik, Campus E1.4, 66123 Saarbrücken, Germany
{dylla, miliaraki}@mpi-inf.mpg.de

Abstract. Over the past decade, the two research areas of *probabilistic databases* and *probabilistic programming* have intensively studied the problem of making structured probabilistic inference scalable, but—so far—both areas developed almost independently of one another. While probabilistic databases have focused on describing tractable query classes based on the structure of query plans and data lineage, probabilistic programming has contributed sophisticated inference techniques based on knowledge compilation and lifted (first-order) inference. Both fields have developed their own variants of—both exact and approximate—top-k algorithms for query evaluation, and both investigate query optimization techniques known from SQL, Datalog, and Prolog, which all calls for a more intensive study of the commonalities and integration of the two fields. Moreover, we believe that *natural-language processing* and *information extraction* will remain a driving factor and in fact a longstanding challenge for developing expressive representation models which can be combined with structured probabilistic inference—also for the next decades to come.

Keywords: Probabilistic databases, probabilistic programming, natural-language processing, information extraction.

1 Introduction

Over the past decade, the two fields of *probabilistic databases* and *probabilistic programming* have studied similar problems but developed almost independently of one another. Although this can be explained by a different focus of the two communities (probabilistic databases focus on querying large databases, whereas probabilistic programming has focused largely on learning statistical models from data), there are ample opportunities for a fruitful cross-fertilization between these two domains. Datalog, for example, the logical query language used in databases, lies at the very intersection of relational algebra (which forms also the semantic basis of SQL) and pure

logic-programming languages such as Prolog. Probabilistic programming is closely related to statistical relational learning [27, 56] and today encompasses a rich variety of formalisms including functional languages, such as Church [28] and IBAL [52], and logical ones, such as Prism [62], ICL [53] and ProbLog [12, 22, 39]. Throughout the present paper we shall—within these families—largely focus on probabilistic extensions of logic programming languages such as ProbLog, which is motivated by the close relationship to database query languages such as Datalog.

With this research statement, we specifically motivate for studying the commonalities between the two worlds of probabilistic databases and probabilistic programming. While probabilistic databases benefit from a mature database infrastructure for *data storage* and *query answering* (including various forms of index structures, support for updates, multi-user concurrency control and transaction management), the field of probabilistic programming has intensively studied advanced techniques for probabilistic inference such as *knowledge compilation* and *lifted inference*. In particular state-of-the-art techniques for lifted inference (i.e., the task of inferring probabilities of queries and shared query components from their first-order representation) have so far not been applied in the context of probabilistic databases. Our aim is to focus on integrating their techniques for query processing and probabilistic inference, both at the propositional level (by exploiting recent developments for knowledge compilation) and at the first-order representation (thus investigating new techniques for top-k query processing and lifted inference). Moreover, we believe that *natural-language processing* and *information extraction* will remain a major driving factor for structured probabilistic inference also for the next decades to come. Albeit natural language is inherently structured and—at least in most cases—follows well-defined grammatical rules, an abundance of ambiguities arises in resolving this structure and in detecting the actual meaning of statements expressed in natural language. While these ambiguities are only partly due to shortcomings of current tools, many statements expressed in natural language remain hard to resolve even for humans and often allow for a variety of interpretations.

2 Probabilistic Databases

Managing uncertain data via probabilistic databases (PDBs) has evolved as an established field of research in the database community in recent years. The field meanwhile encompasses a plethora of applications, which are ranging from *scientific data management*, *sensor networks*, *data integration*, to *information extraction* and *knowledge management* systems [65, 71]. While classical database approaches benefit from a mature and scalable infrastructure for the management of relational data, probabilistic databases aim to further combine these well-studied data management strategies with efficient inference techniques that exploit given independence assumptions among database tuples whenever possible. PDBs adopt powerful query languages from relational databases, including relational algebra, the Structured Query Language (SQL), and logical query languages such as Datalog. Besides the two prevalent types of tuple-independent and block-independent probabilistic database models investigated in the PDB literature [65], PDBs may in fact capture arbitrary correlations among database tuples, and in particular recent PDB developments clearly lie at the intersection of

databases and probabilistic graphical models [36, 37, 63]. The Trio probabilistic database system [46, 72], which was developed at Stanford University back in 2006, was the first such system that explicitly addressed the integration of data management (using SQL as query language), lineage (aka. "provenance") management via Boolean formulas, and probabilistic inference based on the lineage of query answers. In particular, the so-called form of *How-lineage* (see [7] for an overview) captures lineage as propositional formulas that trace which relational operations were involved in deriving each individual query answer—in the sense of "How was this query answer derived?" The underlying data model of Trio, coined Uncertainty and Lineage Database (ULDB) [3], was the first PDB approach that was shown to provide a *closed and complete* probabilistic extension to the relational data model which supports all relational (i.e., SQL-based) operations. These principles directly carry over to Datalog (and logic programming), as it is shown in the following example.

Example 1. Figure 1 (from [18]) depicts a tuple-independent probabilistic database which consists of the extensional relations $Directed$, $ActedIn$, $Category$, and $WonAward$, as well as the database views ν_1–ν_4 (depicted in Datalog notation, explicitly showing also the quantifiers for all variables) which define the intensional relations $KnownFor$, $BestDirector$ and $ActedOnly$. View ν_1, for example, expresses that directors are known for a movie category if they occur in the relation $BestDirector$ together with a movie of that category. Likewise, view ν_2 expresses that actors are known for movies that won a best picture award, but only if they appear together in the $ActedOnly$ relation together with that movie. Evaluating the query $KnownFor(X, Crime)$, thus asking for directors or actors X who are known for $Crime$ movies, over the database tuples and views shown in Figure 1 involves an iterative rewriting of the intensional query predicate $KnownFor$ via the views and the extensional relations. By applying a form of top-down SLD resolution [18], we first observe that the head literals of both ν_1 and ν_2 unify with the query literal and thus rewrite $KnownFor(X, Crime)$ as a Boolean combination of the body literals of the two views, as it is shown in Figure 2. In the following two SLD steps, we resolve the remaining intensional literals $BestDirector(X, Z)$ and $ActedOnly(X, Z)$ via views ν_3 and ν_4 in a similar way. Finally, by binding the distinguished query variable X to the corresponding constants of those tuples that unify with the remaining extensional literals, we obtain the two probabilistic query answers $KnownFor(Coppola, Crime)$ and $KnownFor(Tarantino, Crime)$ with their lineages $((t_2 \wedge t_8) \wedge t_{12})$ and $(t_{10} \wedge\ (t_6 \wedge \neg t_3) \wedge t_{13})$, respectively.

A key in making probabilistic inference scalable to the extent that is needed for modern data management applications lies in the identification of tractable query classes [9, 10] and in the adaptation of known inference techniques from probabilistic graphical models to a relational-data setting [63]. While for specific classes of queries, probabilistic inference can directly be coupled with the relational operations [10, 61], the performance may very quickly degenerate when the underlying independence assumptions do not apply or are too difficult to resolve. Despite the polynomial complexity for the data computation step that is involved in finding answer candidates to probabilistic queries, the confidence computation step, i.e., the actual probabilistic inference, for these answers is known to be of exponential cost already for fairly simple select-project-join

Directed

	Director	*Movie*	p
t_1	Coppola	ApocalypseNow	0.8
t_2	Coppola	Godfather	0.9
t_3	Tarantino	PulpFiction	0.7

ActedIn

	Actor	*Movie*	p
t_4	Brando	ApocalypseNow	0.6
t_5	Pacino	Godfather	0.3
t_6	Tarantino	PulpFiction	0.4

WonAward

	Movie	*Award*	p
t_7	ApocalypseNow	BestScript	0.3
t_8	Godfather	BestDirector	0.8
t_9	Godfather	BestPicture	0.4
t_{10}	PulpFiction	BestPicture	0.9

Category

	Movie	*Category*	p
t_{11}	ApocalypseNow	War	0.9
t_{12}	Godfather	Crime	0.5
t_{13}	PulpFiction	Crime	0.9
t_{14}	Inception	Drama	0.6

$\nu_1 : \forall X, Y\ KnownFor(X, Y) \quad :\!- \exists Z\ BestDirector(X, Z), Category(Z, Y)$
$\nu_2 : \forall X, Y\ KnownFor(X, Y) \quad :\!- \exists Z\ WonAward(Z, BestPicture), ActedOnly(X, Z), Category(Z, Y)$
$\nu_3 : \forall X, Y\ BestDirector(X, Y) :\!- Directed(X, Y), WonAward(Y, BestDirector)$
$\nu_4 : \forall X, Y\ ActedOnly(X, Y) \quad :\!- ActedIn(X, Y), \neg Directed(X, Y)$

Fig. 1. A tuple-independent probabilistic database about movies with a number of extensional and intensional relations.

(SPJ) queries in SQL. Consequently, much of the recent research in PDBs has focused on establishing a *dichotomy of query plans* [9] for which confidence computations are either of polynomial runtime or are $\#\mathcal{P}$-hard [60, 65]. Generally, $\#\mathcal{P}$ denotes an exponential complexity class of counting problems which in the case of a probabilistic databases resolves to $\#\mathcal{SAT}$, i.e., the problem of counting the number of variable assignments that satisfy a Boolean formula.

Thus, efficient strategies for probabilistic ranking and early pruning of low-confidence query answers (see [42] for an overview of various ranking semantics in PDBs) remain a key challenge also for the scalable management of uncertain data via PDBs. Recent work on efficient confidence computations in PDBs has addressed the $\#\mathcal{P}$-hardness of general SQL queries mainly from two ends, namely by restricting the class of queries that are allowed, i.e., by focusing on so-called *safe query plans* [10], or by considering a specific class of tuple-dependencies, commonly referred to as *read-once functions* [64][4]. Intuitively, safe query plans denote a class of queries for which confidence computations can directly be coupled with the relational operators and thus be performed by an extensional query plan. Read-once formulas, on the other hand, denote a class of Boolean lineage formulas which can be factorized (in polynomial time) into a form where every variable in the formula (each representing a database tuple) appears at most once, thus again permitting efficient confidence computations.

While safe plans focus on the characteristics of the query structure, and read-once formulas focus on the logical dependencies among individual data objects, top-k style pruning approaches have recently been proposed as an alternative way to address confidence computations in PDBs [5, 18, 51, 57]. These approaches aim to efficiently identify the top-k most probable answers, using lower and upper bounds for their marginal probabilities, without the need to actually compute the exact probabilities of these answers. Earlier works by Suciu, Dalvi and Ré [57, 5] addressed this by approximating the probabilities of the top-k answers using Monte-Carlo-style sampling techniques. Olteanu and Wen [51], on the other hand, recently extended their approach of decomposing proposi-

[4] These have been shown to be equivalent in [50] for unions of conjunctive queries without self-joins but, in general, read-once formulas may allow for efficient inference for a wider class of queries by considering individual tuple-dependencies.

tional formulas in order to derive a similar (but exact) bounding approach based on finding shared subqueries and by employing partially expanded ordered-binary-decision-diagrams (OBDDs) [24]. In particular the latter top-k algorithm [51] can effectively circumvent the need for exact confidence computations and can still—in many cases—return the top-ranked query answers in an exact way. However, as opposed to top-k approaches in deterministic databases [21, 32, 66], none of the former top-k approaches in PDBs saves upon the data computation step that is needed to find potential answer candidates. Thus, extensive materialization is required for queries with multiple nested subqueries or over multiple levels of potentially convoluted views.

In [18], Dylla et al.—for the first time in the context of PDBs—develop an integrated approach that combines both the data and confidence computation steps needed for intensional query evaluations, i.e., for queries where confidences cannot be computed via an efficient extensional query plan. By introducing a new notion of *first-order lineage formulas*, this approach iteratively computes lower and upper bounds for the marginal probabilities of all query answers that are represented by such a first-order formula. Lineage formulas are derived from a top-down grounding algorithm of queries in Datalog, which incrementally expands the query literals against the views and the extensional relations until the top-k query answers with the highest marginal probabilities are determined. The algorithm supports the full expressiveness of Datalog, including recursion and stratified negation, which (without recursion) forms also the core of relational algebra and SPJ queries in SQL (however without grouping and aggregations, see [23, 25] for extensions required to support full relational algebra in a PDB setting). Figure 2 depicts the construction of such a first-order lineage formula for the example PDB shown in Figure 1, as it is triggered by the query $KnownFor(X, Crime)$. The figure also explicitly depicts the existential quantifiers that are introduced for local variables occurring only in the bodies of the view definitions.

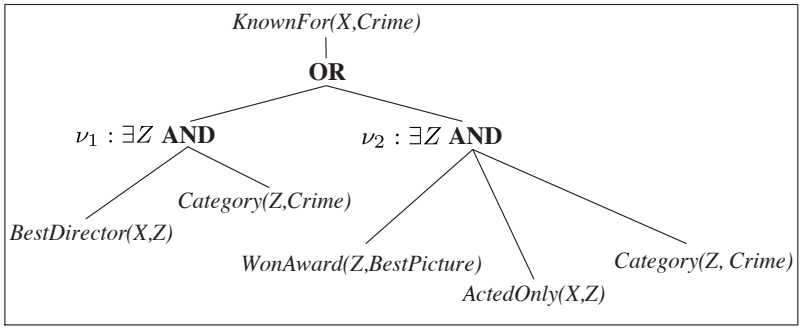

Fig. 2. A partially grounded lineage formula for the query $KnownFor(X, Crime)$.

3 Probabilistic Programming

Probabilistic programming (PP) extends programming languages with probabilistic primitives, thus resulting in rich languages for probabilistic models. Probabilistic logic programming languages based on Prolog, such as Prism [62], ICL [53] and ProbLog [12,

22, 39], have been among the first such extensions studied. These are originally based on Sato's distribution semantics [62] which in turn exhibits surprising similarities to the possible-worlds semantics [1], as it is used in PDBs today. PP approaches combine a logic program with probabilistic facts, and are thus closely related to PDBs that associate probabilities to tuples. In fact, the afore described PDB maps directly to PP and ProbLog, as illustrated in Figure 3, where we have one probabilistic fact for each relational tuple in the tables of the PDB and the same clauses (this time however in Prolog notation) as before.

```
0.8::directed(coppola,apocalypseNow).     0.3::wonAward(apocalypseNow,bestScript).
0.9::directed(coppola,godfather).         0.8::wonAward(godfather,bestDirector).
0.7::directed(tarantino,pulpFiction).     0.4::wonAward(godfather,bestPicture).
                                          0.9::wonAward(pulpFiction,bestPicture).

0.6::actedIn(brando,apocalypseNow).       0.9::category(apocalypseNow,war).
0.3::actedIn(pacino,godfather).           0.5::category(godfather,crime).
0.4::actedIn(tarantino,pulpFiction).      0.9::category(pulpFiction,crime).
                                          0.6::category(inception,drama).

knownFor(X,Y) :- bestDirector(X,Z), category(Z,Y).
knownFor(X,Y) :- wonAward(Z,bestPicture), actedOnly(X,Z), category(Z,Y).
bestDirector(X,Y) :- directed(X,Y), wonAward(Y,bestDirector).
actedOnly(X,Y) :- actedIn(X,Y), \+directed(X,Y).
```

Fig. 3. A ProbLog program equivalent to the PDB in Figure 1.

Despite their closely related semantics, both PDBs and PPs have so far focused on different types of probabilistic queries. In a (probabilistic) database setting, one is typically interested in finding *all solutions* to a query Q, i.e., on generating all query answers, where the aim is to find tuples of constants that serve as substitutions for the distinguished query variables, together with the probability p for each such substitution θ that $Q\theta$ is true [65]. In contrast, in probabilistic programming and many related statistical relational learning settings [27], the focus lies on computing the probability that there *exists a solution* to the query Q, i.e., the probability of $\exists\theta : Q\theta$ [12]. Let us illustrate the different semantics of PDBs and PP by following up on our initial example.

Example 2. Following the ProbLog program shown in Figure 3, the query `knownFor(X, crime)`, thus again asking for directors or actors `X` who are known for `Crime` movies, would produce answers `X` $=$ `coppola` with probability 0.36 and `X` $=$ `tarantino` with probability 0.0972 in a PDB (see Figure 1). In PP, on the other hand, this query would be interpreted to ask for the existence of any directors or actors `X` who are known for `crime` movies, and would thus be answered by the single probability 0.422.

Moreover, in probabilistic programming and statistical relational learning, it is common to compute *conditional probabilities*, thus answering queries of the form $P(Q|E)$, where E denotes the evidence, i.e., particular observations that have been made. A second important inference setting in PP, as in probabilistic graphical models, is maximum-a-posteriori (MAP)—often also referred to as most-probable-explanation (MPE)—inference, where the task is to find the most likely joint truth-value assignment to a set of ground query atoms under a given evidence. PDBs, on the other hand, consider top-k

queries, which ask for the k most likely answer substitutions for a given query, so far however without considering evidence or any form of conditioning [18, 51]. Both the fields of PDBs and PP have developed their own semantics and algorithms for top-k queries over probabilistic data [18, 39, 51, 58], and thus we believe that their combination constitutes a very natural and intriguing subject for further research. Finally, while PP conditions on evidence given at query time, the use of hard logical constraints is common in databases; one then has to condition the probabilities of queries in order to account also for the "impossible worlds" (i.e., those that violate the constraints) [40]. A detailed semantic categorization of these queries (together with their answering mechanisms) thus will likely reveal a unifying framework for query answering and probabilistic inference in both PDBs and PP. Given the closeness in semantics, a key difference between PDBs and PPs, however is that PP languages have the expressive power of a programming language and are, unlike PDBs, Turing equivalent.

In contrast to PDBs, PP so far has mostly focused on general-purpose probabilistic inference techniques, including approaches based on *knowledge compilation*, but did not yet investigate the complexity of evaluating different query structures such as safe plans or read-once formulas. This focus on inference techniques, both exact and approximate, is in part due to the close connection of PP to statistical relational learning and graphical models. In PP, parameters (or probabilities of tuples) are often learned from data rather than provided as part of the model, which typically includes solving large numbers of inference tasks during learning and thus requires efficient inference. Knowledge compilation (see [11] for an overview of techniques) is a standard technique in graphical models and PP [12, 22, 39, 68], but has only fairly recently been considered in PDBs [23, 34, 35, 50]. These approaches aim to compile a Boolean formula into a more succinct representation formalism, over which probabilistic inference can then be performed in polynomial time with respect to the size of the compiled data structure. The work on ProbLog has been the first to use knowledge compilation for inference in a PP setting. It compiles given information about (in)dependencies among individual data objects that contribute to the answer of a probabilistic query into a compact data structure that allows for efficient probability computation. Early work in PP has used OBDDs [12, 39], while the state of the art employs even more succinct structures such as d-DNNFs [22]. While ProbLog uses a single type of target structure that is independent of the query structure, work in PDBs has more explicitly considered different classes of query structures, based on query plans or data lineage, for which probabilistic inference is either tractable or $\#\mathcal{P}$-hard [65]. Thus, there is indeed a close connection between knowledge compilation in PP and the management of lineage formulas in PDBs that has so far not been investigated explicitly.

Probabilistic programs and statistical relational models represent knowledge at an abstract level that makes abstraction of specific (types of) objects and the contexts in which they appear. In this way, it is possible to identify groups of "similar" data objects that can be handled interchangeably. This closely corresponds to finding symmetries in the underlying logical representation. Recent techniques for *lifted inference* take advantage of such symmetries to avoid redundant computations and thus mitigate the cost of inference [68]. The problem of lifted probabilistic inference was first introduced by Poole [54] and has attracted a lot of attention since then (see [13, 45] for some seminal

approaches and [38] for an overview). In PP and statistical relational learning, lifted inference has been well-studied, but in PDBs this is still largely absent (but see [33]). The advantage of lifted inference is that, for certain classes of PP, inference is guaranteed to be polynomial in the size of the domains (i.e., it is "domain-liftable"), whereas non-lifted inference would be exponential. Van den Broeck [14, 67] has contributed the first positive and negative results concerning (domain-)liftability. A state-of-the-art technique for lifted inference is weighted first-order model counting (WFOMC), which is based on the concept of first-order d-DNNF circuits [15]. Such a circuit represents the possible worlds or answers to a given query. A simple example of a first-order formula and the corresponding circuit is shown in Figure 4 (from [67]). The theory states that smokers are only friends with other smokers. The circuit introduces a new domain variable D which denotes a subset of the domain $People$. It states that there exists such a D for which (i) all people in D are smokers, (ii) no other people are smokers, and (iii) smokers are not friends with non smokers. The circuit makes abstraction of the possible subsets D. Only the size of the subset matters for performing probabilistic inference (and hence the circuit captures symmetries between all sets of the same size).

$$\forall X, Y \in People : smokes(X) \wedge friends(X, Y) \Rightarrow smokes(Y).$$

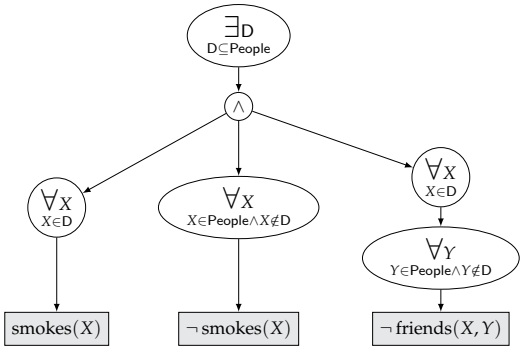

Fig. 4. A first-order formula and its compact representation as a first-order d-DNNF.

By comparing the first-order lineage structure in Figure 2 and the d-DNNF structure in Figure 4, it is clear that the methods developed in both fields are heading in the same direction. Therefore, we expect that the more mature lifted techniques known from PP can help to also accelerate probabilistic inference for first-order lineage formulas in PDBs.

4 Information Extraction

Natural language inherently is highly structured. This simple matter stands in a sometimes surprising contrast to common database jargon, in which natural-language text generally is coined by the term "unstructured data". Any natural-language statement,

such as a piece of text, a sentence, or even just a short phrase—at least when formulated in a meaningful way—follows a well-defined syntactic structure which can be resolved—at least in most cases—by current tagging and parsing tools. Nearly every text snippet, be it an entire Wikipedia article or just a short tweet, contains mentions of named entities, which "just" need to be disambiguated and be put into proper relationships to other entities. Today, a majority of the syntactic variants of statements expressible in natural language are already resolved correctly by current dependency parsers and link grammars, while the remaining cases are often difficult to resolve even for humans. Ambiguities in the parsing structures can be captured—at least to some extent—by probabilistic models such as conditional random fields (CRFs) and probabilistic context-free grammars (PCFGs), which are usually based on hidden Markov models (HMMs) (see [41, 55] for an introduction). State-of-the-art parsers are able to learn the transition probabilities among states and partly even allow for learning the production rules of the underlying grammar from training sentences.

Information extraction (IE) aims to bridge the gap between natural language and structured (typically relational) representation formalisms. Recent, so-called *domain-oriented*, IE approaches [4, 30, 47] extract triplets of facts captured in the popular Resource Description Framework (RDF). Each such RDF fact consists of a pair of known entities (or a pair of an entity and a string literal, respectively) and a canonical relation that connects this pair. Domain-oriented IE approaches operate over a well-defined set of canonical entities and relationships (such as entities represented by Wikipedia articles and relations occurring as infobox attributes) and often require just a handful of fairly simple regular expressions for the extraction step. Conversely, approaches in the field of *open* IE [2, 6] aim to relax the requirement of producing a canonical set of target entities and relations, which generally improves extraction recall but also makes it much harder to integrate their output with structured inference. These approaches employ series of natural-language processing tools, such as part-of-speech (POS) tagging, named-entity recognition (NER), and partly more sophisticated techniques such as semantic role labeling (SRL) [20]. Only few approaches exist that aim to bridge the gap between the two worlds of domain-oriented and open IE by mining for hierarchies of verbal phrases that capture the semantic relationships among entities [48]. Despite its simplicity, RDF will likely not prevail as a representation formalism for natural language. RDF inherently reaches its limitations (or at least quickly loses its conciseness) when capturing different modalities of facts, such as temporal, spatial or other modifiers, or when capturing more than just binary relationships among entities. The ClausIE [8] system, for example, is an interesting approach that directly turns the dependency-based parse trees of natural-language sentences into higher-arity, but sparse, relations. From a more relational perspective, learning logical deduction rules via inductive logic programming (ILP) [26], which again intersects with the field of logic programming, provides a very promising application of structured inference in large and incomplete knowledge bases. Information extraction and scalable probabilistic inference have been addressed by a plethora of approaches in machine learning [17, 43, 44, 49, 59], but only relatively few works so far address the exact interplay of information extraction tools (based on HMMs and CRFs) and inference in probabilistic databases [29, 69, 70]. In summary, we advocate that turning natural language into a machine-readable and processable format (a pro-

cess coined "machine reading" in [19] and "language learning" in [6]) constitutes one of the best showcases for managing uncertain data we have seen so far. This application domain will certainly remain a major challenge for both structured and probabilistic inference also for next decades to come. The integration of domain-oriented and open IE techniques, on the other hand, is a key for applying these inference techniques at Web scale.

5 Conclusions

The methods and technologies developed for probabilistic inference both in databases and in logic programming are becoming mature and scalable and—already today—in many cases allow for exact probabilistic inference over query answers derived from many thousands of variables. Probabilistic databases currently support a wide range of queries formulated in expressive declarative query languages such as SQL, SPARQL, Datalog and function-free Prolog, as well as XPath and restricted subsets of XQuery. Probabilistic programming provides sophisticated knowledge compilation techniques and initial approaches for lifted (first-order) inference, with judiciously tuned approximation algorithms for the cases when exact inference remains intractable. Information extraction will be the driver to foster indexing and querying natural-language contents with temporal, spatial, and other contextual annotations that go beyond just RDF facts. Structured machine learning with scalable probabilistic inference and natural language modeling are two longstanding problems in artificial intelligence and machine learning [16]. Scaling-out probabilistic inference via distributed main-memory platforms for data storage and querying (see, e.g., [43]) will provide major challenges also for future research. Parallel query processing techniques, using both vertical and horizontal partitioning schemes (see, e.g., [31, 73]), have a great potential to serve as a solid database backend for these inference techniques as well. The integration of all of the above will require major contributions from the fields of databases, machine learning, and natural-language processing. It's about time to join our forces!

References

1. Serge Abiteboul, Paris Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. *Theor. Comput. Sci.*, 78(1):159–187, 1991.
2. Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the Web. In *IJCAI*, pages 2670–2676, 2007.
3. Omar Benjelloun, Anish Das Sarma, Alon Y. Halevy, Martin Theobald, and Jennifer Widom. Databases with uncertainty and lineage. *VLDB J.*, 17(2):243–264, 2008.
4. Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - a crystallization point for the Web of Data. *J. Web Sem.*, 7(3):154–165, 2009.
5. Jihad Boulos, Nilesh N. Dalvi, Bhushan Mandhani, Shobhit Mathur, Christopher Ré, and Dan Suciu. MYSTIQ: a system for finding more answers by using probabilities. In *SIGMOD*, pages 891–893, 2005.
6. Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

7. James Cheney, Laura Chiticariu, and Wang-Chiew Tan. Provenance in databases: Why, how, and where. *Found. Trends databases*, 1:379–474, 2009.
8. Luciano Del Corro and Rainer Gemulla. ClausIE: clause-based open information extraction. In *WWW*, pages 355–366, 2013.
9. Nilesh N. Dalvi and Dan Suciu. The dichotomy of conjunctive queries on probabilistic structures. In *PODS*, pages 293–302, 2007.
10. Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.
11. Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17(1):229–264, 2002.
12. Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. In *IJCAI*, pages 2462–2467, 2007.
13. Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. Lifted first-order probabilistic inference. In Lise Getoor and Benjamin Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.
14. Guy Van den Broeck. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *NIPS*, pages 1386–1394, 2011.
15. Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, pages 2178–2185, 2011.
16. Thomas G. Dietterich, Pedro Domingos, Lise Getoor, Stephen Muggleton, and Prasad Tadepalli. Structured machine learning: the next ten years. *Machine Learning*, 73(1):3–23, 2008.
17. Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.
18. Maximilian Dylla, Iris Miliaraki, and Martin Theobald. Top-k query processing in probabilistic databases with non-materialized views. In *ICDE*, pages 122–133, 2013.
19. Oren Etzioni, Michele Banko, and Michael J. Cafarella. Machine reading. In *AAAI Spring Symposium: Machine Reading*, pages 1–5, 2007.
20. Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. Open information extraction: The second generation. In *IJCAI*, pages 3–10, 2011.
21. Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.
22. Daan Fierens, Guy Van den Broeck, Ingo Thon, Bernd Gutmann, and Luc De Raedt. Inference in probabilistic logic programs using weighted CNF's. In *UAI*, pages 211–220, 2011.
23. Robert Fink, Larisa Han, and Dan Olteanu. Aggregation in probabilistic databases via knowledge compilation. *PVLDB*, 5(5):490–501, 2012.
24. Robert Fink and Dan Olteanu. On the optimal approximation of queries using tractable propositional languages. In *ICDT*, pages 174–185, 2011.
25. Robert Fink, Dan Olteanu, and Swaroop Rath. Providing support for full relational algebra in probabilistic databases. In *ICDE*, pages 315–326, 2011.
26. Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, pages 413–422, 2013.
27. Lise Getoor and Benjamin Taskar. *An Introduction to Statistical Relational Learning*. MIT Press, 2007.
28. Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. Church: A language for generative models. In *UAI*, pages 220–229, 2008.
29. Rahul Gupta and Sunita Sarawagi. Creating probabilistic databases from information extraction models. In *VLDB*, pages 965–976, 2006.

30. Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artif. Intell.*, 194:28–61, 2013.
31. Jiewen Huang, Daniel J. Abadi, and Kun Ren. Scalable SPARQL querying of large RDF graphs. *PVLDB*, 4(11):1123–1134, 2011.
32. Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.*, 40:11:1–11:58, 2008.
33. Abhay Kumar Jha, Vibhav Gogate, Alexandra Meliou, and Dan Suciu. Lifted inference seen from the other side: The tractable features. In *NIPS*, pages 973–981, 2010.
34. Abhay Kumar Jha and Dan Suciu. Knowledge compilation meets database theory: compiling queries to decision diagrams. In *ICDT*, pages 162–173, 2011.
35. Abhay Kumar Jha and Dan Suciu. On the tractability of query compilation and bounded treewidth. In *ICDT*, pages 249–261, 2012.
36. Abhay Kumar Jha and Dan Suciu. Probabilistic databases with MarkoViews. *PVLDB*, 5(11):1160–1171, 2012.
37. Bhargav Kanagal and Amol Deshpande. Lineage processing over correlated probabilistic databases. In *SIGMOD*, pages 675–686, 2010.
38. Kristian Kersting. Lifted probabilistic inference. In *ECAI*, pages 33–38, 2012.
39. Angelika Kimmig, Bart Demoen, Luc De Raedt, Vitor Santos Costa, and Ricardo Rocha. On the implementation of the probabilistic logic programming language ProbLog. *Theory and Practice of Logic Programming*, 11:235–262, 2011.
40. Christoph Koch and Dan Olteanu. Conditioning probabilistic databases. *PVLDB*, 1(1):313–325, 2008.
41. John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.
42. Jian Li, Barna Saha, and Amol Deshpande. A unified approach to ranking in probabilistic databases. *PVLDB*, 2(1):502–513, 2009.
43. Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Distributed GraphLab: A framework for machine learning in the cloud. *PVLDB*, 5(8):716–727, 2012.
44. Andrew McCallum, Karl Schultz, and Sameer Singh. FactorIE: Probabilistic programming via imperatively defined factor graphs. In *NIPS*, pages 1249–1257, 2009.
45. Brian Milch, Luke S. Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling. Lifted probabilistic inference with counting formulas. In *AAAI*, pages 1062–1068, 2008.
46. Michi Mutsuzaki, Martin Theobald, Ander de Keijzer, Jennifer Widom, Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Raghotham Murthy, and Tomoe Sugihara. Trio-One: Layering uncertainty and lineage on a conventional DBMS. In *CIDR*, pages 269–274, 2007.
47. Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In *WSDM*, pages 227–236, 2011.
48. Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. Discovering and exploring relations on the Web. *PVLDB*, 5(12):1982–1985, 2012.
49. Feng Niu, Christopher Ré, AnHai Doan, and Jude W. Shavlik. Tuffy: Scaling up statistical inference in Markov Logic Networks using an RDBMS. *PVLDB*, 4(6):373–384, 2011.
50. Dan Olteanu and Jiewen Huang. Using OBDDs for efficient query evaluation on probabilistic databases. In *SUM*, pages 326–340, 2008.
51. Dan Olteanu and Hongkai Wen. Ranking query answers in probabilistic databases: Complexity and efficient algorithms. In *ICDE*, pages 282–293, 2012.
52. Avi Pfeffer. IBAL: A probabilistic rational programming language. In *IJCAI*, pages 733–740, 2001.

53. David Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1-2):7–56, 1997.
54. David Poole. First-order probabilistic inference. In *IJCAI*, pages 985–991, 2003.
55. Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.
56. Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors. *Probabilistic Inductive Logic Programming — Theory and Applications*, volume 4911 of *Lecture Notes in Artificial Intelligence*. Springer, 2008.
57. Christopher Ré, Nilesh N. Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, pages 886–895, 2007.
58. Joris Renkens, Guy Van den Broeck, and Siegfried Nijssen. k-optimal: A novel approximate inference algorithm for ProbLog. *Machine Learning*, 89(3):215–231, July 2012.
59. Sebastian Riedel. Improving the accuracy and efficiency of MAP inference for Markov Logic. In *UAI*, pages 468–475, 2008.
60. Dan Roth. On the hardness of approximate reasoning. *Artif. Intell.*, 82:273–302, 1996.
61. Anish Das Sarma, Martin Theobald, and Jennifer Widom. Exploiting lineage for confidence computation in uncertain and probabilistic databases. In *ICDE*, pages 1023–1032, 2008.
62. Taisuke Sato. A statistical learning method for logic programs with distribution semantics. In *ICLP*, pages 715–729, 1995.
63. Prithviraj Sen, Amol Deshpande, and Lise Getoor. PrDB: managing and exploiting rich correlations in probabilistic databases. *VLDB J.*, 18(5):1065–1090, 2009.
64. Prithviraj Sen, Amol Deshpande, and Lise Getoor. Read-once functions and query evaluation in probabilistic databases. *PVLDB*, 3(1):1068–1079, 2010.
65. Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.
66. Martin Theobald, Gerhard Weikum, and Ralf Schenkel. Top-k query evaluation with probabilistic guarantees. In *VLDB*, pages 648–659, 2004.
67. Guy Van den Broeck. *Lifted Inference and Learning in Statistical Relational Models*. PhD thesis, Informatics Section, Department of Computer Science, Faculty of Engineering Science, Katholieke Universiteit Leuven, January 2013.
68. Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, pages 2178–2185, 2011.
69. Daisy Zhe Wang, Michael J. Franklin, Minos N. Garofalakis, and Joseph M. Hellerstein. Querying probabilistic information extraction. *PVLDB*, 3(1):1057–1067, 2010.
70. Daisy Zhe Wang, Eirinaios Michelakis, Michael J. Franklin, Minos N. Garofalakis, and Joseph M. Hellerstein. Probabilistic declarative information extraction. In *ICDE*, pages 173–176, 2010.
71. Gerhard Weikum and Martin Theobald. From information to knowledge: harvesting entities and relationships from Web sources. In *PODS*, pages 65–76, 2010.
72. Jennifer Widom. Trio: A system for data, uncertainty, and lineage. In *Managing and Mining Uncertain Data*. Springer, 2008.
73. Kai Zeng, Jiacheng Yang, Haixun Wang, Bin Shao, and Zhongyuan Wang. A distributed graph engine for Web scale RDF data. In *SIGMOD*, 2013. to appear.