ABSTRACT
Many data management applications must deal with data which is uncertain, incomplete, or noisy. However, on existing uncertain data representations, we cannot tractably perform the important query evaluation tasks of determining query possibility, certainty, or probability: the problems are hard on arbitrary uncertain input instances. We thus ask whether we could restrict the structure of uncertain data so as to guarantee the tractability of query evaluation. We present our tractability results for tree and tree-like uncertain data, and a vision for probabilistic rule reasoning. We also study uncertainty about order, proposing a suitable representation, as well as uncertain data conditioned by additional observations.

1. INTRODUCTION
Traditional database management theory assumes that data is correct and complete. However, more and more applications deal with incomplete, uncertain and noisy data. For instance, data is extracted or inferred automatically from random Web pages by automated and error-prone extraction programs [14]; integrated from diverse sources through approximate mappings [20]; contributed to collaboratively editable knowledge bases [42] by untrustworthy users; or approximate mappings [20]; contributed to collaboratively editable knowledge bases [42] by untrustworthy users; or deduced from the imprecise answers of random workers on crowdsourcing platforms [9, 36].

Various kinds of uncertainty can hold on the data, which influences our choice of how to represent it. The best known is fact uncertainty: we are dealing with statements for which we do not know whether they are correct or incorrect. However, there are other situations, such as order uncertainty: we are interested in an order relation on facts (e.g., time, relevance) or on the objects (e.g., preference, quality), and we only have partial information about this order (e.g., it was obtained from conflicting user preferences, or by integrating event sequences that are not synchronized).

The straightforward way to extend existing data management paradigms to uncertain data is to represent explicitly all possible states of the data (which we call possible worlds), and to define the semantics of queries as returning all answers that can be obtained on the possible worlds. Of course, this simple scheme is not practical: there are often exponentially many possible worlds, so we cannot represent them all, much less query them. Instead, we must design representation systems, which concisely describe a collection of possible worlds, and evaluate queries directly on the representation, to return a representation of all possible results.

This implies that, in general, the query results will themselves be uncertain. Still, they have many uses. They allow us to determine whether some answers are possible, or certain; or to estimate which ones are likely, based on a probabilistic model on the underlying source of uncertainty (e.g., the trustworthiness of sources). We can also use them to specialize the result of the query, without reevaluating it from scratch, if we ever obtain information that lifts some of the uncertainty. For instance, when we have access to human users (e.g., via the crowd), we can use the uncertain query results to estimate which additional knowledge would help reduce the uncertainty, and ask them the right questions to make the query output more crisp.

Hence, we have defined semantics for uncertain data, but this tells us nothing about whether we can manage it tractably. Sadly, in general, this is not the case. For example, in the context of fact uncertainty, consider the framework of tuple-independent (or TID) instances [33], which are the simplest kind of probabilistic relational instances: all facts are independently present or absent with a given probability. Consider the conjunctive query (CQ) $q : \exists x \exists y R(x, y)S(x, y)F(y)$. It is #P-hard [17] to compute the probability that $q$ holds on an input TID instance, and this is a data complexity result, i.e., it is only in the instance, even when the query is assumed to be fixed. This contrasts with the $AC^0$ data complexity [2] of CQs on traditional instances. In other contexts, e.g., order uncertainty, or uncertain information that has been partly disambiguated using crowd answers, we do not even know whether there are good representation systems.

The goal of my PhD is to address this problem from a theoretical angle, identifying situations where the structure of uncertain data ensures the tractability of query evaluation in terms of possibility, necessity, and probability. In other words, my goal is to show that query evaluation is tractable when we make assumptions on the data: on the structure of the underlying facts, on the kind of uncertainty, and on its structure (e.g., fact correlations). The hope would be to identify tractable classes covering practical examples of uncertain data, and achieve a theoretical understanding of why and how we can tractably query them.

The main focus is on fact uncertainty, which is studied in Section 2. We first study tree representations of data, in the context of probabilistic XML [32], giving examples of how a tractability result for local uncertainty models on trees [15] can be generalized to global uncertainty models where the scopes of uncertain events have bounded overlap. We then move to relational representations, and explain how the XML tractability results generalize to uncertain relational instances that have bounded treewidth, in the sense of having a simultaneous bounded-width decomposition of their underlying instance and their uncertainty annotations. We then describe perspectives to extend this result, the main one being the problem of reasoning under uncertain rules.

My second focus (in Section 3) is on order uncertainty.
discriminately, but are there safe ways to use them without leading to intractability? In [7] we have answered this question in the affirmative, by introducing the notion of event scopes, and stating the first (to our knowledge) non-trivial sufficient condition on PrXML trees ensuring the tractability of query evaluation. Intuitively, the scope of an event is the set of nodes where the value of this event must be “remembered” when trying to evaluate a query on the tree; in Figure 1, the scope of $e_{Jane}$ are the nodes “surname” and “place of birth” and their descendants. The scope of a node $n$ is the set of events having $n$ in their scope. We showed that for PrXML documents where the scope of all nodes have size bounded by a constant, the evaluation of a fixed MSO query can be performed in PTIME in the input document.

In fact, this claim follows from much more general results about structurally tractable instances. We now turn to this.

2.2 Tree-Like Data

When data cannot easily be represented as a tree, a natural way to write it is to use relational databases (or instances) [2]. We can then represent uncertain data using the formalism of c-instances [30, 27], which augments relational instances with propositional annotations on facts using Boolean events, each event valuation defining a possible world obtained by retaining only the facts whose annotation evaluates to true. An example c-instance is given as Table 1, describing which trips should be booked depending on the conferences that a researcher wishes to attend: PODS is taking place in Melbourne and STOC in Portland. We can use pc-instances [27, 29] to model probabilistic distributions on instances, simply by giving independent probabilities to the events of the c-instance.

There are several query languages for relational instances and (p)c-instances: existentially quantified conjunctions of atoms (known as conjunctive queries or CQs), MSO queries, Datalog [2], or some of its variants such as frontier-guarded Datalog [11]. However, we know that evaluating a fixed CQ is already #P-hard in data complexity, even on TIDs [33] which are much less expressive than pc-instances.

Yet, as we saw in the previous section, hardness does not necessarily hold for tree-shaped data. Could we then show the tractability of query answering on TIDs which are assumed to be tree-shaped? In fact, we can show [7] that tractability holds for TID instances of bounded treewidth [39], which intuitively requires that they are close to a tree.

THEOREM 1. Defining the treewidth of a TID as that of its underlying relational instance (forgetting about the probabilities), for input TIDs with treewidth bounded by a constant, the evaluation of a fixed MSO query can be performed in PTIME data complexity. The complexity drops to linear time if we assume constant-time arithmetic operations.

This result cannot directly generalize to pc-tables, because they allow arbitrary propositional annotations on facts, so
CQ evaluation is already \#P-hard in data complexity on single-fact pc-instances. Hence, to cover pc-tables as well, we would need to limit the expressiveness of annotations. Our idea is to write annotations as Boolean circuits rather than formulae, and look at the treewidth of the annotation circuit. We can show [7] that tractability does not follow from bounded treewidth of the instance and of the circuit in isolation; rather, we must require the existence of a bounded-treewidth tree decomposition of the instance and circuit, which respects the link between circuit gates and the facts that they annotate. We call those bounded-treewidth pcc-instances, and we can show:

**Theorem 2.** Evaluating a fixed MSO query on bounded-treewidth pcc-instances has PTIME or linear-time data complexity (depending on the cost of arithmetic operations).

This general result implies Theorem 1 and the tree tractability results of the previous section. It relates to Courcelle’s theorem [16] for usual relational instances, which shows that MSO queries (which are generally NP-hard) can be evaluated in linear-time data complexity if we assume constant treewidth. To show this, one compiles [41] the MSO query \( q \), in a data-independent fashion, to a tree automaton \( A \) which can read tree encodings of bounded-treewidth instances and determine whether they satisfy \( q \). We follow the same approach, but we show that \( A \) can also be run on an uncertain instance \( I \), producing a lineage circuit \( C \) that describes which possible worlds of \( I \) are accepted by \( A \). We then show that \( C \) has bounded treewidth, and so the probability that \( I \) satisfies \( q \) can be computed from \( C \) using standard message passing techniques [34]. Hence, bounded-treewidth pcc-instances are structurally tractable.

Our method relates to CQ evaluation approaches on probabilistic instances which compute a lineage of the query and evaluate the probability of that lineage. This line of related work has proven fruitful, e.g., to identify a dichotomy [18] between safe and unsafe queries (depending on the data complexity of evaluating them on TID instances). Our approach is different: we assume a restriction on the data, namely bounded treewidth, and show that the lineages that we obtain are always tractable, for any query that can be compiled to an automaton: beyond CQs, this covers MSO, frontier-guarded Datalog, and more generally guarded second-order queries. Also, our lineages are circuits rather than formulae, and are constructed from an automaton for the query rather than an execution plan. We use this to cast a new light on semiring provenance: in the case of monotone queries, our lineage circuits are provenance circuits [19] matching standard definitions of semiring provenance [26] for absorptive semirings. We show this by connecting the automaton to a new intrinsic definition of provenance for the query.

Of course, our assumption of bounded-treewidth means that we do not cover many practical use cases, beyond tree-shaped data. We could address this from a theoretical angle, as we do not know yet whether Theorem 2 generalizes to weaker assumptions such as bounded clique-width or hypertree-width [22]. However, in more pragmatic terms, we hope to extend our result to partial tree decompositions: we would structure uncertain instances as a high-treewidth core and low-treewidth tentacles, and evaluate queries by combining Theorem 2 on the tentacles and sampling-based approximate methods on the core. The assumption is that real-world uncertain data, while it may not have bounded treewidth, should have large low-treewidth parts, so that eliminating them with our approach should make query evaluation more efficient overall. A similar idea (in a more restricted context) was recently studied in [35], where it was shown to improve the performance of source-to-target query evaluation on uncertain graphs.

Another point that we intend to study, in terms of practical applicability, is the question of combined complexity. Indeed, compiling MSO queries to automata is generally non-elementary in the query. One possibility around this would be to adapt the construction to monadic Datalog [24]; another one would be to investigate the performance of practical automata compilation techniques [28].

### 2.3 Reasoning Under Probabilistic Rules

We conclude our study of structural tractability for tuple uncertainty, by describing our vision for tractable reasoning under probabilistic rules.

When evaluating queries on incomplete knowledge bases (KBs) such as Wikidata [42], we may miss some answers because the corresponding facts are absent from the KB. However, if we know some hard constraints about the KB (e.g., the “located in” relation is transitive), it makes more sense to say that a query is true if it is certain under the constraints, namely, if it is satisfied by all completions of the KB that obey the constraints. This is called open world query answering.

Our claim is that it would be more useful to reason under soft rules, i.e., probabilistic rules. For instance, if the birth date of a person is missing from the KB, we can deduce a likely range for the date using any other fact about the person. Likewise, a citizen of a country often lives in that country, and probably speaks the official language of the country. Such rules could be produced by association rule mining [3], or using KB-specific methods [21]. Of course, some of the facts that they imply may be wrong, but on average we expect them to help reduce incompleteness in the KB. Hence, we would hope to obtain better query answers by asking for the likely answers under many uncertain rules, rather than the certain answers under a few hard rules.

There are already several approaches to reason under uncertain rules. The first ones are generic probabilistic reasoning frameworks, such as graphical models or Markov Logic Networks [38], and reasoning systems such as ProbLog [37]; but it is not obvious how to encode database rule languages to such settings, hopefully maintaining tractability of reasoning. The second approaches are extensions of Datalog\(^{+/-}\) to probabilistic rules [23], but we do not think they can be used to reason under uncertain rules, because their semantics is different. Indeed, if we say that citizens of a country are born there with 80% probability, their semantics is that the rule is either always true or always false, with probability 80%. Our desired semantics is that the rule applies in 80% of cases on average. Perhaps closest to our approach is [1], but their approach deals only with vanilla Datalog and cannot assert the existence of new elements.

Of course, there are many reasons why our desired semantics for probabilistic rules is hard to formalize. First, there may be multiple independent ways to deduce the same fact, so determining the overall probabilities of new facts is tricky, especially as there may be correlations, and cyclic derivations where facts are deduced via a path that involve themselves. Second, the possible consequences of the rules may be infinite, so that there may be infinitely many possible worlds to consider (unlike, e.g., pc-instances). We hope to formalize such a semantics by a variant of the chase [2], yielding both a probabilistic process to generate possible
worlds, and a reasoning process to describe the possible linkages of facts. Alternatively, another possibility would be to eliminate some rules by rewriting them into the query.

The second challenge posed by probabilistic rules is the question of tractability. For some languages (e.g., guarded Datalog [25] with terminating chase), we hope to preserve treewidth-based tractability guarantees from the instance to the rule consequences. If the chase does not terminate, a possibility would be to represent it as a recursive Markov chain [12], or to truncate it and control the error.

Beyond guarded rules, it would be practically useful to support equality constraints, number restrictions (e.g., “people have at most two parents”), or closed-world domains: for instance, when we deduce that a person has a country of residence, it would be preferable that they have a country, rather than being a fresh null. However, we do not know which distribution to assume on such reuses, and we fear that our criteria for tractability would no longer apply.

3. ORDER UNCERTAINTY

We now leave the standard setting of fact uncertainty and move to order uncertainty: we want to model data where we are unsure about the order between facts or data items. In this setting, to justify the tractability of uncertain data, we need to invent the right representation systems to model the uncertain data and the query output. Of course, uncertain order relations between elements and tuples could in principle be modeled as fact uncertainty, but this would ignore the structure of the uncertainty: it would create many facts and correlations, leaving little hope for tractability.

Yet, there are many scenarios where order uncertainty is specifically needed. For instance, consider the problem of integrating lists of items that are ordered by an unknown criterion, e.g., a proprietary relevance function, or the preferences of various users. If we wish to exist in the knowledge base, rather than being a fresh null. However, we do not know which distribution to assume on such reuses, and we fear that our criteria for tractability would no longer apply.

4. CONDITIONING

Last, we turn to data that has been conditioned [40]: starting with an original uncertain data instance, we have revised it to force the outcome of certain probabilistic events, given new observations or additional information.

The motivation for this kind of uncertain data is very general, because uncertain data can often be made more certain if we are ready to pay the price. For instance, we can often ask a human expert to verify whether a fact is really true, or whether an event occurred or not. If we do so, we must figure out two things: which question to ask, and how do we incorporate the answer to our uncertain model.

Integrating the answer already poses a problem of tractability: for instance, we can easily condition a c-instance to indicate that an event is true, but it is much harder to force the annotation of a fact to be true, as it can be an arbitrary formula. Further, we do not know at all whether structural tractability guarantees on the original instance can be preserved by conditioning.

Choosing the question is an entirely different issue. It is tricky to even define what the best question is, and even harder to find a sensible definition that is tractable to evaluate. The most relevant study of this issue may come from crowd data sourcing: when we try to extract knowledge from a crowd of human users, we are never sure about what we know, because we can never fully trust the answers that have been produced by the crowd workers. Yet, from our current knowledge and our current estimation of the likely answers, we must decide what is the next question that we should ask to the crowd [9], to reduce our uncertainty on the final answer. To our knowledge, however, existing crowd data sourcing techniques [36] use very ad-hoc representations which are specific to some simple query types.

Hence, it is an important challenge to design a generic uncertainty representation framework suitable for such iterative scenarios: at each step, the data is conditioned based on our observations, and we need to choose the queries that we intend to make, relative to the cost of these queries. Besides crowd sourcing, we believe that our vision of such a system [8] applies to many situations that involve a tradeoff between spending more resources and acquiring more knowledge.

5. CONCLUSION

We have presented our results about how to deal with order uncertainty, and fact uncertainty on tree and tree-like
instances. We have presented many perspectives to extend them: for instance, representing the consequences of uncertain deduction rules, or the result of conditioning the existing data with additional information.

There are interesting directions left to explore. An important one would be to evaluate the practical applicability of what we propose, on datasets or for concrete tasks involving uncertain ordered data or low-treewidth data. The design of a practical implementation would also raise theoretical questions: How to combine our methods with approximate methods such as sampling? Which optimizations would help us deal with the high combined complexity?

In terms of representations, we hope to understand how order and fact uncertainty can be combined, and whether the result could be extended to cover more uncertainty types, such as the result of conditioning. Indeed, we believe that an fundamental challenge for uncertain data representation is to support dynamic situations, where the data can evolve: new facts are extracted, deduction rules are fired, and existing information is disambiguated and clarified through human queries or complex processing. Designing such a framework would be both a theoretical and a practical challenge.

6. REFERENCES