Managing Uncertainty and Conflicts in a Distributed World

Serge Abiteboul
INRIA Saclay & ENS Cachan
serge.abiteboul@inria.fr

Daniel Deutch
Tel Aviv University
danielde@post.tau.ac.il

1. THE PROBLEM

We consider a distributed setting where peers in a network exchange information, and apply reasoning to derive further information. We note that uncertainty is common in such setting. Peers may have disagreements and state or infer conflicting facts. Peers can settle conflicts by choosing between contradicting base or inferred facts, which introduces a first cause of uncertainty. Then, there is an inherent uncertainty introduced by the asynchronous environment: the order in which messages are sent and received, as well as the order of applying reasoning steps, are both uncertain.

In this short paper, we consider the problem of *modeling* the dynamics of such networks, accounting for uncertainty. We briefly recall a proposal for the management of uncertainty, namely datalog^{fd} [2]. We consider extending it to the distributed Datalog dialect *Webdamlog* introduced in [1]. We mention preliminary results.

In Section 2, we ignore distribution and focus on a single peer, recalling results from [2]. The distributed case is considered in Section 3.

2. THE CENTRALIZED CASE

The syntax in the centralized setting amounts to datalog, with FDs imposed on intensional relations to model conflicts. We refer to the language as datalog^{fd}. In [2], we have studied two possible semantics for datalog^{fd}. To simplify, we consider here only the set-at-a-time semantics from there. That semantics is in the spirit of previous proposals based on fixpoint logic with a witness operator [3], choice in logic programs [7] or repairs [4]. At each stage, we add a maximal set of immediate consequences that are consistent with the facts that have been inferred so far. Nondeterminism results from the choice of *one consistent set of facts* at each stage. This is continued until a fixpoint is reached.

We measure the non-determinism using probabilities. In the presence of several options for a next derivation step (several instantiations), we consider each option as equiprobable. If a datalog^{fd} program is applied to a database instance, a probabilistic database is obtained.

Example 1 *Suppose that we have a relation $IsIn(x, y, z)$ with the semantics that z believes that a person x is in a location y. We assume that this relation follows the FD $1, 3 \rightarrow 2$, i.e., according to someone, a person is in a single location. Now suppose that Bob believes his friends (this may be encoded as a Datalog rule) and that his friends have conflicting opinions about where is Alice. This introduces some uncertainty of what Bob believes. Indeed, with the semantics of datalog^{fd}, Bob will believe that Alice is, say in London, with the probability equal to the proportion of his friends who believe she is there. Note that this is the case because for every friend we get a distinct derivation option, and the derivation options are equiprobable.*

This language has been studied in [2], and we briefly mention next some main results. We studied the transformations that may be defined by datalog^{fd} programs: we showed that datalog^{fd} with *nsat* semantics captures exactly the known class NDB-PTIME of of queries that can be computed by a non-deterministic Turing Machine in polynomial time. We also considered the representation of the possible worlds induced by queries: We presented a PTIME (data complexity) algorithm for computing a c-table that captures the result of a datalog^{fd} program for each of the two semantics, even when the input is also represented as a c-table. Also, for the probabilistic semantics, we showed a counterpart representation based on pc-tables [9], but only for non-recursive queries with one FD per relation. (The general case remains open.) We have also studied whether the probability of deriving some particular fact could be efficiently computed: The problem is ♯P-hard, but the probability may be efficiently approximated via sampling.

3. THE DISTRIBUTED CASE

Our study of uncertainty and conflicts in the centralized setting was a necessary first step, towards our goal of studying it in presence of distribution. We next briefly recall the Webdamlog language that extends datalog to support a declarative, logical model of inference in presence of distribution. We then account for contradictions by extending datalog^{fd} to this logical model of distribution. Finally, we highlight some results in this respect, as well as remaining gaps. We start by providing a simple example demonstrating some aspects of *Webdamlog*, referring the reader to [1] for details.

Example 2 *We next formalize Example 1 using* Webdamlog *syntax. There is a separate $IsIn$ relation for each peer p, that is denoted by $IsIn@p(\$X, \$Y)$; intuitively "peer p thinks that $\$X$ is in $\$Y$". Additionally peer p has a separate relation $baseIsIn@p(\$X, \$Y)$ that contains the original opinion of the peer. Each peer p includes*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

the rules:

$$IsIn@\$P(\$X, \$Y) \quad :- \quad Friend@p(\$P), IsIn@p(\$X, \$Y)$$
$$IsIn@p(\$X, \$Y) \quad :- \quad baseIsIn@p(\$X, \$Y)$$

Observe the use of variable $\$P$ that matches all the peers that are friends of peer p. Intuitively the first rule says that if you know where someone is, you let your friends know.

The semantics of Webdamlog with probabilities, i.e., Webdamlog^{fd}, is defined as follows. A *run* consists of an infinite sequence of moves of the various peers. At each step, a peer is randomly selected. This peer computes locally a fixpoint. This results in deriving some facts that enrich the peer locally, and other facts that are sent to other peers. For the semantics locally to a peer, we use the semantics of datalog^{fd}. In absence of additional information, we assume that at each step, each peer has equal probability of being activated. One advantage of this semantics is that it gives us the expected *voting* semantics in the distributed counterpart of Example 1: the probability of inferring a fact at a peer is the fact relative support among his friends.

The computational problems previously mentioned have interesting counterparts in the distributed setting. We illustrate them briefly by considering two issues: (i) the construction of a compact (exact) representation of a query answer, (ii) a sampling technique to compute an *approximation* of a query answer.

Compact Representation. A standard approach for query evaluation in presence of non-determinism is to compute a compact representation of all possible query results. In particular, c-table [10] (and their probabilistic counterpart pc-table [9]) representation may usually be obtained by keeping the full provenance [8, 6, 5] of the possible computations. Such representations could serve as the exact result for query evaluation in our settings, capturing possible query answers along with their probabilities. Towards the goal of obtaining such c-tables, we must assume all peers are willing to fully disclose all their information, in particular the provenance of facts. Observe that keeping track of provenance is not trivial, because of the possibly unbounded interaction between the peers.

We start by presenting a construction that "almost" works, in the sense that it indeed constructs a sequence of c-tables that capture more and more precisely the set of possible worlds, but unfortunately the exact representation is never reached. By induction, a c-table representing the set of possible worlds after i steps is constructed. For this, at step i, we introduce probabilistic events for the next move. For instance, some events would correspond to the selection of the peer to move, and others to the choice of an FD ordering, Unfortunately, convergence is not guaranteed for this sequence of c-tables.

To fix this construction, we must not introduce events for moves that do not change the state. If at step i, we can move from state σ to state σ', $\sigma' \neq \sigma$, we introduce an event $e_{i,\sigma,\sigma'}$. Now the computation of the c-table at step i to step $i+1$ is much more complicated, but possible (it is a lengthy case analysis). The difference is that the c-table at step i represents the possible states after i "real" moves, i.e., i changes of states. So we are now guaranteed to converge because of the inflationary nature of the computation. (To check convergence, we have to verify if the tables at step i and $i+1$ are equivalent, which can be performed.)

This construction presents significant drawbacks. First, the size of the representation that is obtained may be prohibitively large. Even more importantly, there are privacy issues. Using this approach, all peers must disclose fully their base facts and also their rules, indicating how they derive new facts. Whether there is a way to avoid this, and still achieve some reasonable form of representation is an interesting challenge.

Distributed Sampling. We introduce a technique for computing query probabilities that is based on sampling and does not require a peer to disclose all its information. The main challenge is in achieving coordination between the peers to obtain the samples and in guaranteeing convergence for arbitrary order of peer activation. We assume that there are finitely many peers and that each peer knows about the existence of all other peers and can contact them. Then, some peer p_q performs one round of sampling as follows:

1. p_q asks the other peers to make a copy of their initial state in temporary relations. Then the peers simulate a run using these temporary relations as follows.

2. p_q asks each peer in the system whether it can move. If some peer can move, then p_q uniformly chooses the next peer to be activated out of those than can move. The peer that is activated makes a probabilistic move and then returns control to p_q for the next simulation step (goto 2.). If no peer can move (a fixpoint has been reached), then p_q records that one more sample has been obtained and whether q is answered positively in this sample and moves on (goto 3.).

3. If the desired number of samples has not been reached, p_q initiates a new sample (goto 1.); and if it has, p_q terminates by giving an estimate of the probability that q holds.

We can show that the sampling algorithm is effective: first, the expected time to obtain each sample using the algorithm is polynomial in the size of the input instance; second, the number of samples required for obtaining the correct probability up to an additive error of ϵ, with probability at least δ, is $O(\frac{ln(\frac{1}{\delta})}{\epsilon^2})$.

4. CONCLUSION

This paper describes on-going research. More results have been obtained that could not be shown here due to space limitations. There are many additional challenges for future work. One particularly intriguing direction is the study of *explanations*, as follows. We have studied in [2] the extension of notions such as fact *influence* [12, 11] to datalog^{fd}. Intuitively, the influence of a fact is measured by its effect on the probability of a query result. Influence analysis in the distributed setting of Webdamlog^{fd} would provide further understanding of the network dynamics.

The language Webdamlog^{fd} captures in a nutshell reasoning issues, faced when dealing with inconsistencies and contradictions in a social network. We believe that it should be an invaluable tool to understand issues such as: how opinions are formed in a network, how rumors are spread, how peers become influential, etc.

Acknowledgments. This work has been supported in part by the Advanced European Research Council grant Webdam on Foundations of Web Data Management, by the Israeli Science Foundation, by the Israeli Ministry of Science, by the US-Israel Binational Science Foundation (BSF) and by the Broadcom Foundation and Tel Aviv University Authentication Initiative.

5. REFERENCES

[1] S. Abiteboul, M. Bienvenu, A. Galland, and E. Antoine. A rule-based language for web data management. In *PODS*, 2011.

[2] S. Abiteboul, D. Deutch, and V. Vianu. Deduction with contradictions in datalog. In *ICDT*, 2014.

[3] S. Abiteboul and V. Vianu. Non-determinism in logic-based languages. *Ann. Math. Artif. Intell.*, 3(2-4):151–186, 1991.

[4] L. Antova, C. Koch, and D. Olteanu. $10^{(10^6)}$ worlds and beyond: efficient representation and processing of incomplete information. *VLDB J.*, 18(5):1021–1040, 2009.

[5] O. Benjelloun, A. Sarma, A. Halevy, M. Theobald, and J. Widom. Databases with uncertainty and lineage. *VLDB J.*, 17, 2008.

[6] P. Buneman, S. Khanna, and W. Tan. Why and where: A characterization of data provenance. In *ICDT*, 2001.

[7] S. Greco, D. Saccà, and C. Zaniolo. Datalog queries with stratified negation and choice: from p to d^P. In *ICDT*, pages 82–96, 1995.

[8] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *Proc. of PODS*, 2007.

[9] T. J. Green and V. Tannen. Models for incomplete and probabilistic information. *IEEE D. Eng. Bull.*, 29(1), 2006.

[10] T. Imielinski and W. L. Jr. Incomplete information in relational databases. *J. ACM*, 31(4), 1984.

[11] B. Kanagal, J. Li, and A. Deshpande. Sensitivity analysis and explanations for robust query evaluation in probabilistic databases. In *SIGMOD Conference*, pages 841–852, 2011.

[12] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. The complexity of causality and responsibility for query answers and non-answers. *PVLDB*, 4(1):34–45, 2010.