Towards Approximating Incomplete Queries over Partially Complete Databases (Extended Abstract)

Ognjen Savković[1], Evgeny Kharlamov[2], Werner Nutt[1], and Pierre Senellart[3]

[1] Free University of Bozen-Bolzano, Italy
[2] University of Oxford, United Kingdom
[3] École Normale Supérieure, France

Motivation. Building reliable systems over partially complete data poses significant challenges because queries they send to the available data retrieve answers that may significantly differ from the *real* answers. This may lead to a wrong understanding of the data and the events and processes it describes. This problem is especially critical for analytical systems that aggregate retrieved data since missing answers may significantly change results of analytical computations, e.g., computation of minimal or average values is sensitive to missing values [2,7]. One way to ensure reliability of (analytical) systems over partially complete data is to guarantee that whatever data they touch is *complete* w.r.t. to the real data.

A possible way to model partial data completeness is with *tuple generating dependencies* (*TGDs*) [1] that specify *what* parts of a relation are complete [8–10]:

$$R^a(\bar{t}) \leftarrow R^i(\bar{t}), \phi^i, \tag{1}$$

where R^a is the *available* (or complete) part of the *ideal* R^i (or real) part of R, $\bar{t}$ is a term of size $ary(R)$, ϕ^i is a conjunction of atoms over $\cdot^i$-annotated predicates. Let Π be a finite set of completeness statements as in Equation (1). With D^a (resp. D^i) we denote a database instance of $\cdot^a$-annotated (resp. $\cdot^i$-annotated) predicates. The semantics of Π is defined using pairs (D^i, D^a) where $D^a \subseteq D^i$, i.e., where the available (incomplete) data is a subset of the real data, as follows: $(D^i, D^a) \models \Pi$ iff $D^a \cup D^i \models \Pi$.

The setting Π can be extended by considering constraints on the ideal part of relations and modelled with TGDs:

$$\exists \overline{X} \psi^i(\overline{X}, \overline{Y}) \leftarrow \varphi^i(\overline{Y}, \overline{Z}), \tag{2}$$

where ψ and φ are conjunctive queries over $\cdot^i$-annotated predicates. If Σ is a finite set of constraints as in Equation (2), then the semantics of Π can be extended to account for constraints Σ, that is, $(D^i, D^a) \models (\Pi, \Sigma)$, by considering only those D^i's that satisfy Σ.

Given Π and Σ, a query Q is (Π, Σ)-*complete* if for any pair (D^i, D^a) such that $(D^i, D^a) \models (\Pi, \Sigma)$ we have $Q^i(D^i) = Q^a(D^a)$, where Q^i (resp. Q^a) is obtained from Q by annotating each predicate with $\cdot^i$ (resp. $\cdot^a$). While query completeness is a desirable property, in practice many queries may be incomplete. In these cases we would like to be able to approximate the original query with alternative queries that are as *close* as possible to the original one, but whose answers can be verified to be complete. A natural kind of approximations are those from below, called *query specialisations*, and above, called *query generalisation*. Formally, given queries Q and Q' and a setting (Π, Σ), Q' is a (Π, Σ)-*specialisation* of Q if $Q'^i \sqsubseteq_\Sigma Q^i$, that is, if Q' is contained in Q over any ideal database that satisfies the constraints Σ. We are interested in complete specialisations, and among them in maximal ones. Formally, a query Q' is a (Π, Σ)-*maximal complete specialisation (MCS)* of a query Q, or just MCS when (Π, Σ) is clear, if *(i)* $Q'^i \sqsubseteq_\Sigma Q^i$, *(ii)* Q' is (Π, Σ)-complete, and *(iii)* Q' is maximal in the sense that there is no other (Π, Σ)-complete Q'' such that $Q'^i \sqsubset_\Sigma Q''^i \sqsubseteq_\Sigma Q^i$. Generalisations and maximal complete generalisations can be defined analogously.

The problem of completeness has been studied in [8–10] for settings with a weaker form of constraints. In particular, it is known that the complexity of checking query completeness ranges from NP for the setting without constraints to Π_2^P for the setting with finite domains. Moreover, approximation has not been studied in the context of partially complete data. In this work we investigate the problem of completeness for conjunctive queries for expressive constraints Σ and the problem of approximation. We now give an overview of our results.

Query Completeness. We prove characterisations of query completeness in terms of the well-studied problem of query containment over TGDs. That is, Q is (Π, Σ)-complete iff $\Pi \cup \Sigma \models (Q^i \subseteq Q^a)$. Interestingly, also the converse holds: the containment under TGDs can be represented as completeness. Query containment under TGDs is known to be undecidable, thus checking completeness is undecidable. Since the undecidability comes from the constraints, we turn our attention to settings with practically motivated types of constraints: (cyclic) foreign keys, acyclic TGDs [6] sticky TGDs [5], guarded TGDs [4]. For all these constraints we show that the combined complexity of completeness is high, at least PSpace-complete.

Query Specialisation. Intuitively, one can specialise a conjunctive query by instantiating the query variables or by joining new atoms. One can do it by following the TGDs of Π and Σ backwards, and thus instantiate and add atoms as little as needed.

More formally, one can find specialisations by a procedure that is similar to the *resolution proof-scheme* [5] or *backward chaining* [3]. More precisely, for each query atom one has to find in $\Pi \cup \Sigma$ a TGD that can transfer the atom or an instantiation. In this way, the atom may need to be instantiated according to the TGD, but it may also mean that one needs to add new atoms from the body of the TGD. For newly introduced atoms now again one has to find their TGD, etc. The difference with the backward chase is that the query that is specialised is the database and the query at the same time. Thus, with each backward application of the TGD one may instantiate the atom but also change the database. This produces a (potentially infinite) set of (potentially infinite) queries that includes all MCSs but also non-maximal specialisations. However, it contain all MCSs. This is because some combination of rules may lead to more general specialisations than others. We also observe that a query may have more than one MCS both among infinite and finite conjunctive queries. Checking if a query Q' is a (Π, Σ)-specialisation of a query Q is undecidable for unrestricted Σ and this corresponds to the case when the procedure above does not terminate. Weak acyclicity of inverted TGDs from Σ (that is, where the direction of the arrow is reversed) yields termination and for such settings each conjunctive query has a finite number of finite size MCSs. It remains an open question if for sticky or guarded inverted Σ we can have a terminating procedure.

We are still working on the problem of generalisation for incomplete queries.

Acknowledgements This was was partially supported by the EPSRC projects MaSI^3, DBOnto, and ED^3, and by the projects MAGIC and PARCIS, funded by the Free University of Bozen-Bolzano.

References

1. Serge Abiteboul, Richard Hull, and Victor Vianu. Foundations of databases. Addison-Wesley, 1995.
2. Marcelo Arenas, Leopoldo Bertossi, Jan Chomicki, Xin He, Vijay Raghavan, and Jeremy Spinrad. Scalar aggregation in inconsistent databases. *Theor. Comput. Sci.*, 296(3), 2003.
3. Jean-François Baget, Michel Leclère, and Marie-Laure Mugnier. Walking the decidability line for rules with existential variables. KR'10, pages 466–476. AAAI Press, 2010.
4. Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.
5. Andrea Calì, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012.
6. Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *Proc. ICDT*, pages 207–224, 2002.
7. Paolo Guagliardo and Leonid Libkin. Making sql queries correct on incomplete databases: A feasibility study. PODS '16, 2016.
8. A.Y. Levy. Obtaining complete answers from incomplete databases. In *Proc. VLDB*, pages 402–412, 1996.
9. Werner Nutt, Sergey Paramonov, and Ognjen Savkovic. Implementing query completeness reasoning. In *CIKM*, pages 733–742, 2015.
10. Simon Razniewski and Werner Nutt. Completeness of queries over incomplete databases. *PVLDB*, 4(11):749–760, 2011.