Ontology-Based Query Answering with Existential Rules
Marie-Laure Mugnier

To cite this version:
1 Framework and objectives

Ontology-Based Query Answering (OBQA) is currently a problem that receives a lot of attention both from knowledge representation and databases communities. The aim is to answer queries that are at least as expressive as conjunctive queries while taking an ontology into account. This is important in order to improve the quality of query answering and interoperability between different sources of data.

The mainstream formalism to deal with ontologies is description logics (DLs) ([Baader et al., 2007]). While historical DLs are very expressive, most of the OBQA research focus on recently introduced lightweight DLs (EL [Baader, 2003] and DL-Lite [Calvanese et al., 2007]). Real-world ontologies expressed in these DLs already exist, such as the medical ontology SNOMED-CT (based on EL).

During my Ph.D, I am considering an alternative, rule-based formalism. I use existential rules [Baget et al., 2011b] (rules for short, also known as Tuple Generating Dependencies [Abiteboul et al., 1994], Datalog+/- [Calvi et al., 2009]), which despite their simple syntactic form are very expressive. Moreover, they allow for a smooth integration to database systems since, in contrast to DLs, they allow for any predicate arity and support variable cycles. Indeed, their form is:

$$\forall x \forall y (B[x,y] \rightarrow \exists z H[y,z])$$

where B (resp. H) is an arbitrary conjunction of atoms called the body (resp. the head) of the rule. The OBQA problem is thus formalized as follows: given two facts (existentially closed conjunctions of atoms) $F$ and $Q$, a set of rules $R$, does $F, R \models Q$ hold? (where $|-$ denotes logical entailment). However, the ability to create new unknown individuals, via existentially quantified variables, makes reasoning extremely complex: the OBQA problem is undecidable, even with a single rule with a single binary predicate ([Baget et al., 2011b]).

A lot of work has been done in the last years in order to define decidable classes of rules, aiming at a good tradeoff between expressivity and complexity. The interested reader can consult [Mugnier, 2011] for a survey of such classes. Some of these class cover the lightweight DLs used for OBQA.

Algorithms for OBQA can be split in two big categories: either they materialize, that is, they use $R$ to infer new data from $F$, or not. When for any fact $F$, this materialization yields a finite fact entailing all possible consequence of $F$ and $R$, $R$ is called a finite expansion set (fes). When this fact is not finite, but has a bounded treewidth, $R$ is said a bounded treewidth set (bts)$^2$. Both conditions ensures decidability of OBQA, but none are recognizable. Non-materializing algorithms perform query rewriting: the initial query $Q$ is rewritten into a query $Q'$, such that $Q$ is entailed by $F, R$ iff $Q'$ is entailed by $F$. When $Q'$ is a finite union of conjunctive queries (UCQs), $R$ is said a finite unification set (fus). Rules translating DL-Lite ontologies are both bts and fus, while rules translating EL are only bts.

The aim of my Ph.D thesis is to identify expressive decidable classes, study the complexity of reasoning for these classes, and design efficient algorithms in the sense that they improve state of the art algorithms.

2 Contributions

In my Ph.D work, I consider both materialization-based and query rewriting approaches. Natural brute-force algorithms exist for fes and fus, but no such algorithm is known for bts. In a joint work with J.-F. Baget, M.-L. Mugnier and S. Rudolph, we have defined an abstract class, named greedy bounded treewidth set (gbts) and provided a worst-case optimal algorithm for it. This class is a subset of bts that covers most of the known recognizable bts classes of rules. Slight adaptation of this algorithm makes it also optimal for these subclasses. This work has been published in [Baget et al., 2011a; Thomazo et al., 2012; Thomazo, 2012].

Pure query rewriting approaches suffer from the exponential blow-up of the size of rewritings w.r.t. to the query, even with solely class or role hierarchies. UCQs are then too large to be efficiently dealt with by RDMS. In [König et al., 2012], we show that this is inherent to UCQs by characterizing the smallest rewriting using that shape of formulas. In [Thomazo, 2013], I use semi-conjunctive queries, which are a more general form of positive existential formula. I present an algorithm for computing such rewritings, and experimentally evaluate their quality by checking the efficiency of evaluation of such queries. First results show that this approach is more efficient than using UCQs.

---

$^1$Constraints and equality rules are also needed, but this is out of the scope of this summary.

$^2$See [Baget et al., 2011b] for formal definition of these classes.
3 Related work

Recognizability of set of rules suited for a materialization-based approach relies on two main criteria: guardedness and acyclicity. Guarded rules (where an atom of the body contains all variables of the body) have been generalized in several ways [Calvi et al., 2009; Baget et al., 2011b; Krötzsch and Rudolph, 2011].

Weak-acyclicity [Fagin et al., 2005] has been generalized into super-weak acyclicity [Marnette, 2009] and join acyclicity [Krötzsch and Rudolph, 2011]. An incomparable notion relies on the notion of rule dependency [Baget et al., 2011b]. [Grau et al., 2012] proposes a semantic condition of acyclicity that generalizes all these notions.

Last, the combined approach mixes materialization and query rewriting: it both extends the data by applying rules (independently of the query) and rewrite the queries with respect to the rules (independently of the data). This has proven to be useful for both DL-Lite and \( \mathcal{EL} \) ontologies.

Query rewriting approaches are applicable in particular to linear [Calvi et al., 2009; Baget et al., 2011b] and sticky [Gottlob et al., 2011] rules, which are fus. Several algorithms have been implemented for linear rules or DL-Lite ontologies, rewriting either into a union of conjunctive queries ([Pérez-Urbina et al., 2009; Gottlob et al., 2011; Venetis et al., 2012; Chortaras et al., 2011]) or into a Datalog program (Presto [Rosati and Almatelli, 2010]). All fus are processed by the algorithm presented in [König et al., 2012]. The question of finding polynomial cases has also been addressed [Kikot et al., 2011].

References


