The Two Views on Ontological Query Answering

Sebastian Rudolph

Technische Universität Dresden, Germany
sebastian.rudolph@tu-dresden.de

Abstract. We attempt a structured view at the ontological query answering problem by distinguishing between two antagonistic perspectives: The knowledge representation perspective considers the ontology as a part of the specified knowledge whereas the database perspective assumes it to be part of the query. These two perspectives give rise to two computation strategies: (data-driven) forward chaining and (query-driven) backward chaining, based on which different types of decidability criteria can be defined. We give an overview of the two views as well as ensuing conditions for decidability, focusing on existential rules as ontological formalism that has lately gained a lot of renewed interest from both the KR and DB communities.

1 Introduction

Intelligent methods for search and management of large amounts of knowledge require a firm theoretical basis blending paradigms and approaches from databases and knowledge representation. It has been widely acknowledged that the task of retrieving information from a body of knowledge via querying can greatly benefit from a mediating logical layer between the factual data and the query. The *ontological query answering* framework, phrased in terms of formal logic, is to check the entailment $D \models_\Sigma Q$, where D, called the *database*, is a set of ground facts, Σ, called the *ontology*, is a set of sentences of some logic, and Q, the *query*, is a logical sentence. The logical formalisms used to express Σ and Q may differ, they are referred to as *ontology language* and *query language*, respectively. In words, the task is to find out if some statement Q follows from D given the background knowledge Σ. One can distinguish between two perspectives differing in whether Σ is considered as part of the query or belonging to the data. These two viewpoints are sketched in Fig. 1.

The Knowledge Representation View. This view is characterized by the idea that D provides only incomplete information about the state of affairs and that Σ serves the purpose of enriching this information with logical consequences of $D \cup \Sigma$, before querying it via Q. The combination of D and Σ is then often referred to as *knowledge base* and is viewed as integrated and condensed description of a domain of interest. A prominent example for this view are description logics (DLs [2]) where D is referred to as *ABox* whereas Σ is called *TBox*.[1] Until not long ago, the query languages used in DL research were restricted to {**false**} ((un)satisfiability checks) or ground facts – until

[1] Some advanced description logics separate Σ into TBox and RBox, depending on the type of logical sentences.

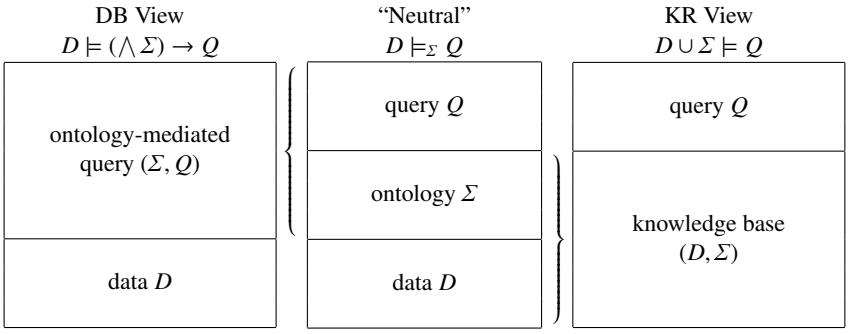

Fig. 1. The two views on ontological query answering.

today inferencing problems of such restricted type are denoted DL *standard reasoning tasks*; only over the last two decades, attention has shifted toward settings where Q can be a conjunctive query.

The Database View. This perspective conceives Σ as part of the query, enhancing the query language's expressivity. In the plain version of this setting, Σ is supposed to be conservative over D: there is a separation into predicates occurring in D (also called *extensional database predicates*, short EDBs) and those only occurring in Σ (referred to as *intensional database predicates*, short IDBs), where the latter are thought to not carry meaning independent of the actual query, rather they are auxiliary predicates, used to store intermediate results toward the computation of the query. In particular, Σ should not allow for inferring new facts over EDB predicates from D. In this scenario, the inferencing task can be rewritten into $D \models \forall \mathbf{P}.(\bigwedge \Sigma \to Q)$, where $\mathbf{P}$ is a sequence of all IDB predicates in Σ viewed as second-order variables. The most widespread example of this view are Datalog queries.

The two perspectives give rise to two fundamentally different approaches for solving the query answering problem. The KR view calls for a transformation of the knowledge base (D, Σ) into a more explicit representation, whereas the DB view suggests the same for the ontology-mediated query (Σ, Q).

In the following, we will see how these two antagonistic generic strategies can be instantiated. As a key showcase, we will use *existential rules*, which will be introduced next.

2 Existential Rules

This section introduces the framework of *existential rules* which are also known under many other names, including tuple-generating dependencies (TGDs), Datalog±, and $\forall\exists$-rules.

Definition 1. *An* existential rule *(or simply* rule *in the context of this paper) is a first-order formula of the form*

$$\forall \boldsymbol{x}.(B_1 \wedge \ldots \wedge B_k \to \exists \boldsymbol{y}.H_1 \wedge \ldots \wedge H_l),$$

where $B_1, \ldots, B_k, H_1, \ldots, H_l$ *(with* $k \geq 0$ *and* $l \geq 1$*) are atoms all of whose variables are in the scope of some quantifier, and where no variable occurs more than once in* $\boldsymbol{x}, \boldsymbol{y}$*. A* Datalog rule *is a rule with no existential quantifiers. A rule with* $k = 0$ *is called a* fact *(a conclusion that is unconditionally true), a set of facts is also referred to as a* database*.*

The premise of a rule is called the body *while the conclusion is called the* head*.*

The rule language hereby introduced is a syntactic fragment of first-order predicate logic (FOL), and we consider it under the according semantics.

Definition 2. *Let* Σ *be a set of rules. A* Boolean conjunctive query *(BCQ) is a formula* $\exists \boldsymbol{v}.Q$ *where* Q *is a conjunction of atoms and* $\boldsymbol{v}$ *contains all variables in* Q*. A BCQ* $\exists \boldsymbol{v}.Q$ *is entailed by* Σ *if it is entailed under standard FOL semantics.*

Checking BCQ entailment for unrestricted existential rules is undecidable [14,8] even with very strong restrictions on the vocabulary or the number of rules [4]. Therefore, a large body of work has been devoted to the identification of restricted rule languages which retain decidability and still allow for sufficient expressiveness.

Still, the existential rules have nice model-theoretic properties which often allow for more efficient reasoning compared to first-order logic in general: the existence of canonical models. For every database D and set of existential rules Σ there exists a *universal model* $\mathcal{I}$, i.e., $\mathcal{I}$ satisfies $\mathcal{I} \models D$, $\mathcal{I} \models \Sigma$, and for each model $\mathcal{J}$ of D and Σ there exists a homomorphism from $\mathcal{J}$ to $\mathcal{I}$. Note that there may be several universal models which are not isomorphic to each other, but there is always a "smallest" one (more formal: one for which identity is the only endomorphism), which is unique up to isomorphism and normally called the *core*.

Universal models are particularly useful when considering entailment of logical formulae whose validity is preserved under homomorphisms, that is, formulae φ for which $\mathcal{I} \models \varphi$ implies $\mathcal{J} \models \varphi$ if a homomorphism from $\mathcal{I}$ to $\mathcal{J}$ exists. For first-order logic, the Łos-Tarski-Lyndon Theorem states that every such formula can be expressed by a positive existential sentence, and consequently by a union of BCQs.

In short: for existential rules and (unions of) Boolean conjunctive queries, query answering (an entailment problem) boils down to model checking in a universal model. Thus, computing some representation of a universal model is a central approach for reasoning with existential rules.

3 KB View – Database Rewriting

Referring back to the described views on the query answering problem, we saw that the "KR view" conceives the ontology as part of the (incomplete) data(base), meant to enrich the latter with further information. This view suggests an inferencing approach which iteratively computes the (possible) factual consequences of the ontology, thus making the knowledge base more explicit and the database less incomplete. Another way to illustrate such a strategy is to view D as a partial model, which may violate some of Σ and is consequently "repaired" by adding new information. This may not only

require adding new facts about known domain elements but it might also necessitate to add new domain elements whose existence can be inferred.

A very general example for such a "model explication" approach is the semantic tableau method in FOL [9,33], which has also been used in many other logics, most notably DLs [3,21]. It is important to note that the way in which the model is "repaired" is non-deterministic in the general case and may in fact lead to different models.

Luckily, the situation is better for existential rules, for which the repair strategy is known as the *chase*, introduced by Maier et al. [27] and extended to query containment by Johnson et al. [23]. Intuitively the chase procedure starts with a given set of factual data (ground facts) and "applies" rules in a production rule, forward-chaining style by introducing new domain elements whenever required by an existentially quantified variable in a rule head. In this setting, the model repairs are deterministic up to renaming and lead toward a canonic model; the only non-determinism left is which instance of which rule to pick for the next "repair step" and how exactly applicability of rules is defined; depending on the particular strategy, different types of the chase have been described (such as oblivious vs. non-oblivious chase, Skolem chase, core chase).

For arbitrary existential rules, termination of the chase cannot be guaranteed, and an infinite set of new domain elements and facts may be created. In general, the chase can thus only serve as semi-decision procedure for query entailment.

It is therefore natural to ask for conditions, under which the chase (or some modification of it) can be used as a decision procedure. In fact, many of the decidable existential rule fragments come about by establishing properties about the chase they create. Thereby, one is normally interested in properties which can be established independently from the underlying database.

3.1 Chase Finiteness through Acyclicity

Finiteness of the (core) chase (for every possibly database D) is a straightforward criterion for ensuring decidability, and rule sets with this property are also called *finite extension sets* [4]. Moreover, finiteness of the core chase occurs exactly if a finite universal model exists. This criterion is undecidable in general (whence it is referred to as an *abstract criterion*), but several sufficient conditions on rule sets guaranteeing chase-finiteness have been identified. Pure Datalog (also known as *full implicational dependencies* [14] or *total TGDs* [8]) is an immediate example, as no new domain elements are created at all and for a finite set of domain individuals only finitely many facts can be created.

A group of *concrete criteria* (that is, criteria which can be syntactically checked) for chase finiteness aims at analyzing the ways how the execution of some rule may trigger executability of other rules. If there is the possibility that, due to cyclic such triggering relationships, more and more domain elements might have to be added when computing the chase, non-termination may arise. The group of criteria aiming at excluding such cyclic dependencies is referred to as *acyclicity conditions*. One example of this is (weak) acyclicity [17,18] which was subsequently refined into *joint acyclicity* [25]. Another approach, pursuing a similar goal by different means is to require acyclicity of the *graph of rule dependencies* [5]. A comprehensive overview of the existing acyclicity notions is provided by Grau et al. [20].

3.2 Chase Treeishness through Guardedness

A more relaxed condition than finiteness of the chase is that the (possibly infinite) chase enjoys a variant of the bounded treewidth property, a notion originating from graph theory but easily transferrable to databases. In words, a (possibly infinite) database D has treewidth of n, if n is the minimal number such that one can come up with an auxiliary structure (called the *tree decomposition*) which is a (possibly infinite) tree whose nodes (called *bags*) are sets containing at most $n + 1$ domain elements from D such that (1) whenever D contains some atom $p(\boldsymbol{d})$, there must be a bag containing all elements from $\boldsymbol{d}$ and (2) for every domain element d from D, the substructure induced by all bags containing d is a tree. Intuitively, the treewidth is a measure of the "tree-ishness" of the database: the lower the treewidth, the more tree-like the database. In particular, tree-shaped databases have treewidth 1.

Now, a rule set is called a *bounded-treewidth set* [4] if there is such a treewidth bound n for its core chase on any database D (where n may depend on D). Decidability of BCQ entailment for bounded-treewidth sets follows from known decidability results for first-order logic theories with the bounded treewidth model property [15]. Again, it is undecidable if a given rule set has this property, but it is possible to come up with sufficient criteria which can be checked easily. Of course, as every finite-extension set is a bounded-treewidth set, all acyclicity conditions would be sufficient criteria. But there is another type of criteria, which covers cases where the chase turns out to be infinite. These criteria are referred to as *guardedness* conditions.

Guardedness (first introduced for full first-order logic [1]) requires that all or certain of the universally quantified variables of a rule appear together in a single "guard" atom in the rule body. Requiring a guard for every universally quantified variable leads to the notion of plain *guarded* rules [10]. Requiring a guard only for variables that also appear in the head (the so-called *frontier* of the rule) yields *frontier-guarded rules* [4,6]. Both notions can be generalized by not requiring guards for variables that cannot possibly represent existentially introduced elements. This idea has been used to arrive at *weakly guarded rules* [10] and *weakly frontier-guarded rules* [4,6].

It turns out that all the above notions of guardedness have a nice computational property: in the course of constructing the chase, the tree-decomposition can be built up simultaneously in a greedy way. It turns out that if a rule set has this property (in which case it is called a *greedy bounded-treewidth set*), BCQ answering can be implemented by a generic worst-case-optimal algorithm that works on the level of the tree-decomposition and computes a finite representation of the full (infinite) chase. This is achieved by detecting repetitions, such that the full tree decomposition can be represented by a finite part of it plus some information how to unfold it into the full infinite structure [34].

3.3 Combining Acyclicity and Guardedness

The principles of acyclicity and guardedness employ different strategies to ensure the desired chase properties. Acyclicity avoids indefinite repetitions of creation of new domain elements whereas guardedness allows to create new atoms involving known domain elements only if these elements already occur together in an atom, thereby preventing that formerly unconnected individuals are arbitrarily linked together by newly

created atoms. The different variants of guardedness presented above already indicate that the plain guardedness criterion can be relaxed in several ways, since not all variables in a rule are equally "dangerous": only those variables that have the potential of triggering the creation of new domain individuals need to be guarded. In fact, this criterion can be further relaxed, drawing from acyclicity ideas: variables are only dangerous if they can trigger the creation of infinitely many new domain individuals through cyclic execution (in which case they are called *glut variables*), whereas variables bringing about only finitely many new elements can still be considered well-behaved. It turns out that it suffices to guard only the glut variables to obtain a bounded-treewidth set. *Glut-guarded* and *glut-frontier-guarded* rules thus defined [25] subsume all known notions of bounded-treewidth sets and are among the most expressive existential rules formalisms, at the cost of high computational complexity of BCQ answering.

4 DB View – Query Rewriting

When taking the perspective of treating the ontology as part of the query, a straightforward strategy is to try and transform this ontology-mediated query into a more explicit representation (using a simpler query language). There is a wide variety of possible target query languages for such a transformation. Figure 2 depicts some of the most popular ones, together with two new ones that were recently introduced as favorable trade-off between expressivity and computational intricacy [30].

A rather basic example for a query-rewriting approach is SLD resolution used in logic programming, where a query is rewritten in all possible ways by factoring in the rules of the logic program in a backward-chaining way. In the case of existential rules, a similar strategy can be applied, with the difference that, unlike in SLD resolution, it cannot be performed query atom by query atom. Rather, so-called pieces (groups of atoms which are connected via variables) have to be rewritten together [5].

4.1 Rewriting into (Unions of) Conjunctive Queries

This piece-based rewriting technique produces in every step a union of Boolean conjunctive queries, which are subsumed by the original ontology-mediated query (that is, whenever one BCQ of the union has a match in a database, the original query matches as well). Moreover, if this step-wise computation stabilizes, it also subsumes the original query and we have arrived at a *perfect rewriting* of the original query into a union of BCQs. Thus, dually to the finite chase condition, one can define *finite unification sets* as rule sets Σ where this rewriting procedure applied to (Σ, Q) terminates for arbitrary BCQs Q [4]. For such rule sets we also obtain a sub-polynomial AC_0 data complexity for BCQ entailment checking. Again, recognizing finite unification sets is undecidable, and various decidable sublanguages are known. Examples include *atomic-hypothesis rules* and *domain restricted rules* [4], *linear Datalog*$^{\pm}$ [11], *sticky sets of TGDs*, and *sticky-join sets of TGDs* [12,13].

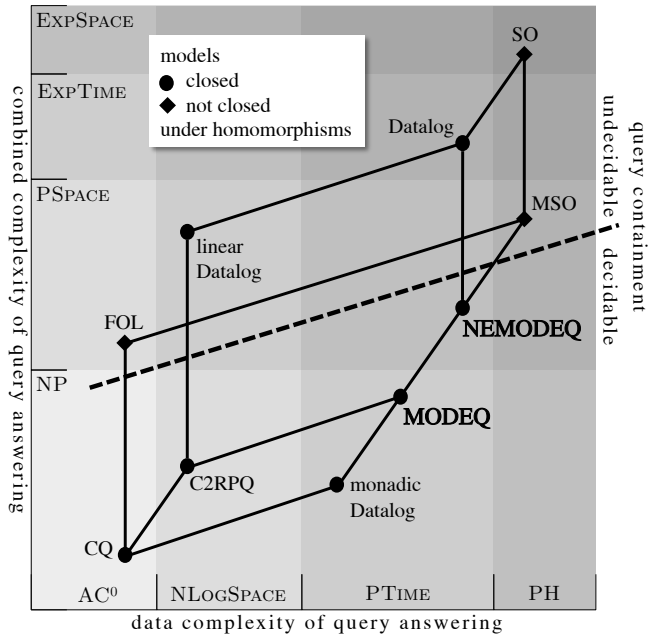

Fig. 2. Overview of complexities and relations of monadically defined queries (MODEQS) and nested MODEQS (NEMODEQS) to other query formalisms; information indicated for conjunctive queries (CQ) and conjunctive 2-way regular path queries (C2RPQ) also hold when allowing unions (UCQ and UC2RPQ); all other formalisms are closed under unions.

4.2 Rewriting into Datalog

Unfortunately, rewriting into queries expressible in first-order logic is not always possible. In fact, most of the known existential rule fragments have a data complexity beyond AC_0, which proves that they cannot be first-order rewritable.

Since considerably more of the known existential rule fragments have PTime data complexity, and Datalog is a well-established querying formalism supported by highly optimized tools, several rewritings into Datalog have been proposed [6,7,19]. Moreover, Datalog-rewritings have been proposed for Horn Description Logics of varying expressivity [26,24,28]. For non-Horn Description Logics, whose data complexity is typically coNP-hard, rewriting into Datalog is not possible, assuming P≠NP. Instead, rewritings into disjunctive Datalog have been developed [22,31].

The fact that most known formalisms with PTime data complexity allow for a rewriting into Datalog may give rise to the hope that **any** existential rule fragment with PTime data complexity might be Datalog rewritable. Such a general finding would also resonate well with the fact that the result is easy to prove for databases with a little bit of extra information: a linear order on the elements. While this would have been a very desirable result, it turns out not to be the case. The following was shown via a pumping argument for Datalog derivations:

Proposition 1 ([16]). *Let Q be the Boolean query matching any database D with two distinguished elements s and t and one binary predicate, which represents a directed graph that either has a cycle or is an acyclic graph having a path from s to t of length $2^{(2^{m^2})}$ for some natural number m. Then, Q is expressible in first-order logic with a least fixed point operator (and hence computable in* PTime*), but cannot be expressed as a Datalog query.*

Our negative result now follows from the existence of an existential rule query that expresses Q using an appropriate Turing machine simulation.

4.3 Rewriting into Well-behaved Datalog fragments

While rewriting into Datalog queries is applicable to many logical fragments, Datalog queries are much more difficult to handle than (unions of) conjunctive queries when it comes to certain tasks which are important for database management. In particular, checking query containment is known to be undecidable for Datalog queries [32]. It is therefore desirable to not use full Datalog as the target query language of rewriting but rather to restrict to fragments of it who are known to be computationally more "well-behaved". We recently identified such query languages which are both expressible in Datalog and in monadic second-order logic while subsuming expressive query formalisms such as monadic Datalog queries and conjunctive 2-way regular path queries (cf. Fig. 2) and showed that they can be used for query rewriting in the presence of rule sets featuring recursive joins, which have turned out to be difficult to handle by other approaches [30].

5 The Mixed View

In some cases the ontology Σ turns out to be structured in a way that Σ as a whole is not suited for any of the two presented approaches, but it can be partitioned into two parts Σ_1 and Σ_2 such that the knowledge base (D, Σ_2) allows for answering queries of some type into which (Σ_1, Q) can be rewritten. This situation is depicted in Fig. 3.

Typically, this requires that Σ_1 is "on top of" Σ_2 applying some notion of stratification. For example, this stratification can be characterized in terms of conservative extensions or – in case Σ_1 and Σ_2 are sets of existential rules – via rule dependencies [5]. This approach allows to establish decidability for a variety of different settings, such as

- Q being a conjunctive query, Σ_1 being a finite unification set, Σ_2 being a bounded-treewidth set [5]
- Q being a conjunctive query, Σ_1 being rewritable into homomorphism-preserved monadic second-order queries and Σ_2 being a bounded-treewidth set [30],
- Q being a conjunctive 2-way regular path query (a nontrivial generalization of conjunctive queries) and Σ being a Horn-$\mathcal{SROIQ}$ knowledge base[2] [29]. As it turns out, every such Σ can be represented as $\Sigma_1 \cup \Sigma_2$, where Σ_1 (the so called RBox)

[2] $\mathcal{SROIQ}$ is the very expressive Description Logic underlying the OWL 2 DL ontology language and Horn-$\mathcal{SROIQ}$ is its Horn fragment.

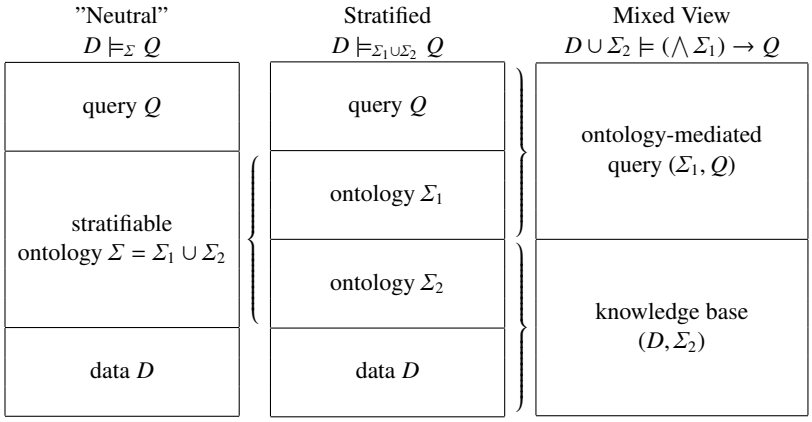

Fig. 3. The mixed view on ontological query answering.

and Q can be rewritten into another conjunctive 2-way regular path query Q' to be executed against the knowledge base (D, Σ_2) (where D and Σ_2 are referred to as ABox and TBox, respectively, in description logic terms).

6 Conclusion

We have reviewed the state of the art in ontological query answering, while focusing on the particular case of existential rules. We argued that the currently available approaches can be roughly separated into two large groups (and combinations thereof) one rewriting the data, the other rewriting the query. While this already helps getting a clearer picture of the field, the ultimate goal would be a joint characterization, capturing these two classes by means of a common criterion. Hopefully, this would allow for defining more expressive, decidable querying frameworks.

Acknowledgements

Much of the synopsis provided in this paper is based on joint research with many colleagues to which I am deeply indebted, most notably Jean-François Baget, Georg Gottlob, Markus Krötzsch, Marie-Laure Mugnier, Magdalena Ortiz, Mantas Šimkus, and Michaël Thomazo. Moreover, I am very grateful for enlightening discussions with Michael Benedikt, Diego Calvanese, Bruno Courcelle and Detlef Seese.

References

1. Andréka, H., Németi, I., van Benthem, J.: Modal languages and bounded fragments of predicate logic. J. of Philosophical Logic 27(3), 217–274 (1998)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edn. (2007)

3. Baader, F., Sattler, U.: An overview of tableau algorithms for description logics. Studia Logica 69(1), 5–40 (2001)
4. Baget, J.F., Leclère, M., Mugnier, M.L.: Walking the decidability line for rules with existential variables. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) Proc. 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'10). pp. 466–476. AAAI Press (2010)
5. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: Extending decidable cases for rules with existential variables. In: Boutilier, C. (ed.) Proc. 21st Int. Joint Conf. on Artificial Intelligence (IJCAI'09). pp. 677–682. IJCAI (2009)
6. Baget, J.F., Mugnier, M.L., Rudolph, S., Thomazo, M.: Walking the complexity lines for generalized guarded existential rules. In: Walsh, T. (ed.) Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). pp. 712–717. AAAI Press/IJCAI (2011)
7. Bárány, V., Benedikt, M., ten Cate, B.: Rewriting guarded negation queries. In: Proc. of MFCS' 13. pp. 98–110 (2013)
8. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Even, S., Kariv, O. (eds.) Proc. 8th Colloquium on Automata, Languages and Programming (ICALP'81). LNCS, vol. 115, pp. 73–85. Springer (1981)
9. Beth, E.W.: Semantic entailment and formal derivability. Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde (18), 309–342 (1955)
10. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Brewka, G., Lang, J. (eds.) Proc. 11th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'08). pp. 70–80. AAAI Press (2008)
11. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: Paredaens, J., Su, J. (eds.) Proc. 28th Symposium on Principles of Database Systems (PODS'09). pp. 77–86. ACM (2009)
12. Calì, A., Gottlob, G., Pieris, A.: Advanced processing for ontological queries. Proceedings of VLDB 2010 3(1), 554–565 (2010)
13. Calì, A., Gottlob, G., Pieris, A.: Query answering under non-guarded rules in Datalog+/-. In: Hitzler, P., Lukasiewicz, T. (eds.) Proc. 4th Int. Conf. on Web Reasoning and Rule Systems (RR 2010). LNCS, vol. 6333, pp. 1–17. Springer (2010)
14. Chandra, A.K., Lewis, H.R., Makowsky, J.A.: Embedded implicational dependencies and their inference problem. In: Proc. 13th Annual ACM Symposium on Theory of Computation (STOC'81). pp. 342–354. ACM (1981)
15. Courcelle, B.: The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. Information and Computation 85(1), 12–75 (1990)
16. Dawar, A., Kreutzer, S.: On Datalog vs. LFP. In: Proc. 35thInt. Colloquium on Automata, Languages and Programming, Part II. pp. 160–171. Springer (2008)
17. Deutsch, A., Tannen, V.: Reformulation of XML queries and constraints. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) Proc. 9th Int. Conf. on Database Theory (ICDT'03). LNCS, vol. 2572, pp. 225–241. Springer (2003)
18. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theoretical Computer Science 336(1), 89–124 (2005)
19. Gottlob, G., Rudolph, S., Šimkus, M.: Expressiveness of guarded existential rule languages. In: Proc. 33rd Symposium on Principles of Database Systems (PODS'02). ACM (2014), to appear
20. Grau, B.C., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity notions for existential rules and their application to query answering in ontologies. J. of Artificial Intelligence Research 47, 741–808 (2013)
21. Horrocks, I., Sattler, U.: A tableau decision procedure for *shoiq*. J. Autom. Reasoning 39(3), 249–276 (2007)

22. Hustadt, U., Motik, B., Sattler, U.: Data complexity of reasoning in very expressive description logics. In: Kaelbling, L., Saffiotti, A. (eds.) Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05). pp. 466–471. Professional Book Center (2005)
23. Johnson, D.S., Klug, A.: Testing containment of conjunctive queries under functional and inclusion dependencies. In: Proc. 1st Symposium on Principles of Database Systems (PODS'82). pp. 164–169. ACM (1982)
24. Krötzsch, M.: Efficient inferencing for OWL EL. In: Janhunen, T., Niemelä, I. (eds.) Proc. 12th European Conf. on Logics in Artificial Intelligence (JELIA'10). LNAI, vol. 6341, pp. 234–246. Springer (2010)
25. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: Walsh, T. (ed.) Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). pp. 963–968. AAAI Press/IJCAI (2011)
26. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable rules for OWL 2. In: Sheth, A., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) Proc. 7th Int. Semantic Web Conf. (ISWC'08). LNCS, vol. 5318, pp. 649–664. Springer (2008)
27. Maier, D., Mendelzon, A.O., Sagiv, Y.: Testing implications of data dependencies. ACM Transactions on Database Systems 4, 455–469 (1979)
28. Ortiz, M., Rudolph, S., Simkus, M.: Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) Proc. 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'10). pp. 269–279. AAAI Press (2010)
29. Ortiz, M., Rudolph, S., Simkus, M.: Query answering in the horn fragments of the description logics SHOIQ and SROIQ. In: Walsh, T. (ed.) Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11). pp. 1039–1044. AAAI Press/IJCAI (2011)
30. Rudolph, S., Krötzsch, M.: Flag & check: Data access with monadically defined queries. In: Hull, R., Fan, W. (eds.) Proc. 32nd Symposium on Principles of Database Systems (PODS'13). pp. 151–162. ACM (2013)
31. Rudolph, S., Krötzsch, M., Hitzler, P.: Type-elimination-based reasoning for the description logic shiqbs using decision diagrams and disjunctive datalog. Logical Methods in Computer Science 8(1) (2012)
32. Shmueli, O.: Decidability and expressiveness aspects of logic queries. In: Proc. 6th Symposium on Principles of Database Systems (PODS'87). pp. 237–249. PODS '87, ACM (1987)
33. Smullyan, R.M.: First-order logic. Dover books on mathematics, Dover (1968)
34. Thomazo, M., Baget, J.F., Mugnier, M.L., Rudolph, S.: A generic querying algorithm for greedy sets of existential rules. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) Proc. 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'12). pp. 96–106. AAAI Press (2012)