Datalog+/-: Questions and Answers

Datalog+/–: Questions and Answers

Georg Gottlob and **Thomas Lukasiewicz** and **Andreas Pieris**

Department of Computer Science, University of Oxford, UK
firstname.lastname@cs.ox.ac.uk

• *What is Datalog$^\pm$ and what are its advantages?*

Datalog$^\pm$ is a rule-based formalism that combines the advantages of logic programming in Datalog with features for expressing ontological knowledge and advanced data modeling constraints. Datalog$^\pm$ provides a uniform framework for query answering and reasoning with incomplete data. It generalizes disparate other KR formalisms such as plain Datalog, description logics (DLs), in particular, DL-Lite and $\mathcal{EL}$, F-Logic Lite, highly relevant fragments of UML class diagrams, data-exchange formalisms, graph query languages such as SPARQL, and so on. Datalog$^\pm$ is a conceptually very simple and clear cut formalism that extends plain Datalog with features such as existential quantifiers, equalities, and the falsum ($\perp$) in rule heads and, at the same time, restricts the rule syntax so as to achieve decidability and, when required, tractability. Its closeness to logic programming allowed us to enrich it by non-monotonic negation under the stratified, well-founded, and stable model semantics.

• *How do Datalog$^\pm$ rules look like?*

A self-explanatory set of Datalog$^\pm$ rules follows:

$$\begin{array}{lcl}
\sigma_1 & : & emp(X) \ \rightarrow \ \exists Y\, hasMgr(X,Y), emp(Y), \\
\sigma_2 & : & hasMgr(X,Y), emp(X) \ \rightarrow \ emp(Y), \\
\sigma_3 & : & emp(X), emp(Y) \ \rightarrow \ colleagueOf(X,Y), \\
\sigma_4 & : & emp(X) \ \rightarrow \ \exists Y\, worksIn(X,Y), dept(Y).
\end{array}$$

Such rules are also known as *tuple-generating dependencies (tgds)*. All variables that are not existentially quantified are assumed to be universally quantified. We just omit the universal quantifiers in front of such rules. As will be pointed out later, Datalog$^\pm$ rules may also contain equalities and the falsum symbol ($\perp$) in their heads.

• *Which reasoning tasks are considered with Datalog$^\pm$?*

The main reasoning task is query answering under the so-called *certain-answers semantics*. If $Q(\mathbf{X})$ is a conjunctive query (CQ) or a union of conjunctive queries (UCQ) with free variables $\mathbf{X}$, D a database over a domain (universe) Δ whose tuples are interpreted in the usual way as ground facts, and Σ a Datalog$^\pm$ program (a.k.a. Datalog$^\pm$ ontology), then the answer to Q consists of all those tuples $\mathbf{a}$ of Δ-elements such that $D \cup \Sigma \models Q(\mathbf{a})$. For Boolean queries, the answer is, accordingly, *true* or *false*. Other relevant reasoning tasks are instance checking, query containment and consistency (or satisfiability) checking. These latter tasks are, however, easily reduced to Boolean query answering.

• *Which are the main decidable Datalog$^\pm$ languages?*

There are so far three main paradigms which extend plain Datalog with existential quantifiers in rule heads, and also guarantee the decidability of query answering:

Weakly-acyclic Datalog$^\pm$, based on weakly acyclic tgds introduced in the context of data exchange (Fagin et al. 2005), guarantees the existence of a finite universal model, which in turn implies the decidability of query answering. The rules $\{\sigma_2, \sigma_3, \sigma_4\}$ given above form a weakly-acyclic set of Datalog$^\pm$ rules. Extensions of weak-acyclicity have been studied, e.g., in (Marnette 2009; Grau et al. 2013).

Guarded Datalog$^\pm$ ensures the existence of treelike universal models, which in turn implies the decidability of query answering (Calì, Gottlob, and Kifer 2013). A rule is guarded if it has an atom which contains all the body-variables; e.g., the rules $\{\sigma_1, \sigma_2, \sigma_4\}$. An important subclass of guarded Datalog$^\pm$ is linear Datalog$^\pm$, where rules have only one body-atom, e.g., the rules $\{\sigma_1, \sigma_4\}$. Extensions of guardedness have been studied in (Baget et al. 2011).

Sticky Datalog$^\pm$ guarantees the termination of backward resolution, and thus the decidability of query answering (Calì, Gottlob, and Pieris 2012). The key idea underlying stickiness is that the body-variables which are in a join always are propagated (or "stick") to the inferred atoms; e.g., the set of rules $\{\sigma_1, \sigma_3, \sigma_4\}$. The main goal of stickiness was the definition of a language that allows for joins in rule-bodies, which are not always expressible via guarded rules.

Interestingly, the consolidation of the above paradigms leads to more expressive decidable formalisms. Notable examples are *glut-guardedness* (Krötzsch and Rudolph 2011), obtained by combining weak-acyclicity and guardedness, *weak-stickiness* (Calì, Gottlob, and Pieris 2012), obtained by joining weak-acyclicity and stickiness, and *tameness*, obtained by combining guardedness and stickiness (Gottlob, Manna, and Pieris 2013). Another important language is *shy* Datalog$^\pm$ (Leone et al. 2012).

• *What about equalities and the falsum ($\perp$) in rule heads?*

Weakly-acyclic Datalog$^\pm$ can be safely combined with arbitrary equality rules, a.k.a. *equality-generating dependen-*

Language	Data Complexity	Combined Complexity
Weakly-acyclic Datalog$^{\pm}$	PTIME-complete UB: (Fagin et al. 2005, Cor. 4.3) LB: (Dantsin et al. 2001, Thm. 4.4)	2EXPTIME-complete UB: implicit in (Fagin et al. 2005) LB: (Calì, Gottlob, and Pieris 2012, Thm. 5.1)
Guarded Datalog$^{\pm}$	PTIME-complete UB: (Calì, Gottlob, and Lukasiewicz 2012, Thm. 6) LB: (Calì, Gottlob, and Lukasiewicz 2012, Thm. 6)	2EXPTIME-complete UB: (Calì, Gottlob, and Kifer 2013, Thm.6.1) LB: (Calì, Gottlob, and Kifer 2013, Thm.6.1)
Sticky Datalog$^{\pm}$	in AC_0 UB: (Calì, Gottlob, and Pieris 2012, Thm. 3.5)	EXPTIME-complete UB: (Calì, Gottlob, and Pieris 2012, Thm. 3.3) LB: (Calì, Gottlob, and Pieris 2012, Thm. 3.4)

Table 1: Complexity of query answering; UB and LB stands for upper and lower bound, respectively.

cies (egds), without destroying decidability of query answering (Fagin et al. 2005). However, this is not true for the other languages mentioned above. For those languages, to guarantee decidability, one has to somewhat restrict the interaction between egds and existential rules, for instance, by admitting *non-conflicting* egds only, see (Calì, Gottlob, and Lukasiewicz 2012; Calì, Gottlob, and Pieris 2012). The falsum ($\perp$) in rule heads can be easily treated since such rules (a.k.a. negative constraints) can be conceived as queries. Thus, a language which guarantees the decidability of query answering can be safely extended with this feature.

• *What is the complexity of query answering under the main Datalog$^{\pm}$ languages?*

The data complexity (the query and the set of rules are considered fixed) and combined complexity of query answering under the main Datalog$^{\pm}$ languages (enriched with equality and the falsum) are summarized in Table 1. In each cell, we have indicated where to find the results.

• *What are typical applications of Datalog$^{\pm}$?*

As already said, Datalog$^{\pm}$ is a framework for knowledge representation and reasoning, and for a variety of other applications. Three of its basic applications are (1) reasoning over conceptual models; (2) ontology querying; and (3) querying the semantic web. Let us give some more details for each of the above applications:

1. UML class diagrams (UCDs) are a widely adopted formalism for modeling the intensional structure of a software system. A fundamental property of UCDs is consistency, showing that the system is realizable in practice. Guarded Datalog$^{\pm}$ is powerful enough to capture a core fragment of UCDs, and in particular each diagram which is the result of a reverse engineering process. Thus, consistency of re-engineered UCDs, a key problem in software engineering, can be reduced to reasoning under guarded Datalog$^{\pm}$. This fact allows us to exploit mature and efficient database technology for solving the consistency problem of re-engineered UCDs, which sensibly improves the performance of existing approaches (Gottlob, Orsi, and Pieris 2014).

2. Description Logics (DLs) are a family of knowledge representation formalisms. It is well-known that reasoning under expressive DLs is very hard or even undecidable. For this reason, several lightweight DLs have been proposed in the literature. Two of the most prominent lightweight DLs are DL-Lite$_{\mathcal{R}}$ (Calvanese et al. 2007) and $\mathcal{EL}$ (Baader, Brandt, and Lutz 2005), which provide the logical underpinning for the OWL 2 QL and OWL 2 EL profiles of OWL 2, respectively. Interestingly, guarded Datalog$^{\pm}$ provides a unifying framework for the above DLs without losing tractability of query answering. More precisely, query answering under DL-Lite$_{\mathcal{R}}$ or $\mathcal{EL}$ can be reduced to query answering under guarded Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2012). Notice that guarded Datalog$^{\pm}$ is strictly more expressive than the above DLs. Sticky Datalog$^{\pm}$ is also more expressive than DL-Lite$_{\mathcal{R}}$ (Calì, Gottlob, and Pieris 2012).

Another example of a knowledge representation formalism which is captured by a slightly more expressive version of guarded Datalog$^{\pm}$, namely weakly-guarded Datalog$^{\pm}$, is F-Logic Lite (Calì, Gottlob, and Kifer 2013). F-Logic Lite (Calì and Kifer 2006) is an expressive restricted version of F-Logic, a well-known formalism, originally introduced for object-oriented deductive databases, but is now most frequently used for expressing Semantic Web ontologies.

3. The problem of querying RDF data is a central issue for the development of the Semantic Web. The query language SPARQL 1.1 was recently released, which includes entailment regimes for RDFS and OWL vocabularies, and a mechanism to express navigation patterns through regular expressions. However, there are still some useful navigation patterns that cannot be expressed in SPARQL 1.1, and the language lacks of a general mechanism to express recursive queries. With the aim of filling this gap, a query language called TriQ, based on weakly-guarded Datalog$^{\pm}$, has been recently proposed (Arenas, Gottlob, and Pieris 2014). In particular, if we focus on the OWL 2 QL profile of OWL 2, then every SPARQL query enriched with the above features can be naturally translated into a query expressed in TriQ, and also in a lightweight version of TriQ, called TriQ-Lite, which ensures tractability of query evaluation. Notice that TriQ captures EXPTIME; the upper bound has been established in (Arenas, Gottlob, and Pieris 2014), while the lower bound in (Gottlob, Rudolph, and Simkus 2014).

• *Apart from the modeling features mentioned above, what else is allowed in the existing Datalog$^{\pm}$ languages? For example, do you allow for negation in rule bodies?*

The most common semantics for nonmonotonic normal (logic) programs have been extended to Datalog$^{\pm}$ with negation in rule bodies. More precisely, we have studied guarded Datalog$^{\pm}$ with negation under the perfect model semantics (PMS) for the stratified case (Calì, Gottlob, and Lukasiewicz 2012), the well-founded semantics (WFS) (Gottlob et al. 2012a; Hernich et al. 2013), and the stable model semantics

(SMS) (Gottlob et al. 2014b). Actually, it is not difficult to define a negation semantics for Datalog$^\pm$, as is this inherited from its standard counterpart in normal logic programs with function symbols. In fact, existentially quantified variables can be replaced by Skolem terms in rule heads. The tricky part, however, is to show that Datalog$^\pm$ with negation under the new semantics remains decidable, and pinpoint the exact complexity of query answering.

The WFS comes in two flavors depending on whether the unique name assumption (UNA) is applied to Skolem terms or not. The first approach leads to the "equality-friendly" WFS (Gottlob et al. 2012a), while the second one leads to the standard WFS (Hernich et al. 2013). The SMS (Gelfond and Lifschitz 1988) is another predominating semantics for nonmonotonic normal programs. In (Gottlob et al. 2014b), we have defined and studied the SMS for guarded Datalog$^\pm$ under the UNA (note that an equality-friendly SMS with UNA also exists). There are cases in which the SMS leads to better query answers than the WFS. For example, given the fact $\mathit{FiveStar}(\mathit{ritz})$ and the set of rules

$$\begin{array}{l}
\mathit{FiveStar}(X) \rightarrow \mathit{Hotel}(X),\\
\mathit{FiveStar}(X), \mathrm{not}\mathit{Pool}(X,Y) \rightarrow \exists Z\, \mathit{Beach}(X,Z),\\
\mathit{FiveStar}(X), \mathrm{not}\mathit{Beach}(X,Y) \rightarrow \exists Z\, \mathit{Pool}(X,Z),\\
\mathit{Beach}(X,Y) \rightarrow \exists Z\, \mathit{SwimOpp}(X,Z),\\
\mathit{Pool}(X,Y) \rightarrow \exists Z\, \mathit{SwimOpp}(X,Z),
\end{array}$$

only the atoms $\mathit{FiveStar}(\mathit{ritz})$ and $\mathit{Hotel}(\mathit{ritz})$ are entailed under the WFS, while the atom $\mathit{SwimOpp}(\mathit{ritz})$ is additionally entailed under the SMS (as desired).

As several DLs can be embedded into Datalog$^\pm$, the decidability results for Datalog$^\pm$ with negation can also be applied to define decidable extensions of DLs with nonmonotonic negation. We have done this in particular for DL-Lite$_\mathcal{R}$ and DL-Lite$_{\mathcal{R},\sqcap}$ (Calvanese et al. 2007), as well as $\mathcal{ELHI}$ (Baader, Brandt, and Lutz 2005). The above example corresponds to the following set of DL axioms, expressed in an extension of $\mathcal{ELHI}$ by nonmonotonic negation:

$$\begin{array}{rcl}
\mathit{FiveStar} & \sqsubseteq & \mathrm{Hotel},\\
\mathit{FiveStar} \sqcap \mathrm{not}\exists \mathit{Pool} & \sqsubseteq & \exists \mathit{Beach},\\
\mathit{FiveStar} \sqcap \mathrm{not}\exists \mathit{Beach} & \sqsubseteq & \exists \mathit{Pool},\\
\exists \mathit{Beach} & \sqsubseteq & \exists \mathit{SwimOpp},\\
\exists \mathit{Pool} & \sqsubseteq & \exists \mathit{SwimOpp},
\end{array}$$

• *Is Datalog$^\pm$ powerful enough for nondeterministic reasoning? In other words, can the existing languages enriched with disjunction in rule heads?*

Weakly-acyclic and guarded Datalog$^\pm$ (and their extensions) can be extended with disjunction, without sacrificing the decidability of query answering. In fact, the impact of disjunction on the complexity of query answering under guarded-based Datalog$^\pm$ languages has been recently investigated in (Gottlob et al. 2012b; Bourhis, Morak, and Pieris 2013). Unfortunately, the above decidability result does not hold for sticky Datalog$^\pm$. A recent (still unpublished) result shows that the combination of stickiness with disjunction leads to undecidability. This result gives rise to the question whether a restricted, yet meaningful fragment of sticky Datalog$^\pm$ can be isolated, which can be safely extended with disjunction. This will be the subject of future research.

• *What other developments have been made related to Datalog$^\pm$?*

1. Query rewriting has been extensively studied. Several algorithms have been designed with the aim of reducing the problem of query answering under Datalog$^\pm$ languages into evaluation of database queries (Gottlob, Orsi, and Pieris 2011; Orsi and Pieris 2011; Gottlob and Schwentick 2012; Gottlob et al. 2014a; Gottlob, Manna, and Pieris 2014a).

2. In (Gottlob et al. 2013), a probabilistic extension of Datalog$^\pm$ that is based on Markov logic networks as underlying probabilistic semantics is presented. It may, e.g., be used in data extraction from the Web.

3. In (Lukasiewicz, Martinez, and Simari 2012), a general framework for inconsistency management in Datalog$^\pm$ ontologies based on incision functions from belief revision is developed. It may, e.g., be used for handling inconsistencies in the Semantic Web.

4. Towards personalized semantic search (e.g., in social networks), in (Lukasiewicz, Martinez, and Simari 2013) an approach to preference-based query answering in ontologies, combining Datalog$^\pm$ with preference management as in relational databases, is presented.

5. Query answering under *finite* models has been also investigated. More precisely, query answering for most of the languages, without egds, mentioned above is *finitely controllable*. This means that the answers to a query remain exactly the same when, instead of considering arbitrary models, we restrict our attention to finite models. Finite controllability of query answering under (weakly-)guarded Datalog$^\pm$ was shown in (Bárány, Gottlob, and Otto 2010) (extending the work by (Rosati 2011)), under sticky Datalog$^\pm$ in (Gogacz and Marcinkowski 2013), and under tame Datalog$^\pm$ in (Gottlob, Manna, and Pieris 2014b).

• *Are there any other modeling features that will be added to Datalog$^\pm$?*

We are planning to investigate more expressive equality assertions and transitivity. Let us give some more details:

1. Though existing languages are powerful enough to express equality assertions, as said, this can only be done as long as the non-conflicting condition is satisfied. However, in real-life examples, new intensional knowledge may be inferred from the equality rules, which implies that the non-conflicting condition is violated. Hence, more general conditions, beyond the non-conflicting one, are needed.

2. Transitivity is partially captured by some Datalog$^\pm$ languages, e.g., weakly-sticky Datalog$^\pm$, as long as the necessary join operations are performed on finitely many values. However, this is not powerful enough for expressing some natural transitivity axioms. We only know that guarded-based Datalog$^\pm$ languages cannot be safely extended with transitivity since this leads to undecidability (Gottlob, Pieris, and Tendera 2013). Thus, the enrichment of Datalog$^\pm$ with this feature will be the subject of future substantial research.

• *Are there any implemented systems for reasoning over Datalog$^\pm$ languages?*

We know of three such systems: (1) Nyaya (Virgilio et al. 2012) is a system able to treat the first-order rewritable fragments of Datalog$^\pm$, that is, linear and sticky Datalog$^\pm$.

In fact, the given set of rules and query are compiled into an SQL query, which is then evaluated over the extensional database; (2) DLV$^{\exists}$ (Leone et al. 2012) implements a bottom-up evaluation strategy for shy Datalog$^{\pm}$ inside the well-known Answer Set Programming (ASP) system DLV (Leone et al. 2006); and (3) Alaska (König et al. 2012), similarly to Nyaya, is able to treat the first-order rewritable fragments of Datalog$^{\pm}$, and it is based on SQL-rewritings. The above are interesting prototypes which show that a complete system, which is able to effectively answer queries over Datalog$^{\pm}$ ontologies, is realistic. The implementation of such a system will be the subject of future research.

Acknowledgements. Work funded by ERC Grant 246858 (DIADEM), and EPSRC grant EP/J008346/1 (PrOQAW).

References

Arenas, M.; Gottlob, G.; and Pieris, A. 2014. Expressive languages for querying the semantic web. In *Proc. of PODS*. To appear.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI*, 364–369.

Baget, J.-F.; Leclère, M.; Mugnier, M.-L.; and Salvat, E. 2011. On rules with existential variables: Walking the decidability line. *Artif. Intell.* 175(9-10):1620–1654.

Bárány, V.; Gottlob, G.; and Otto, M. 2010. Querying the guarded fragment. In *Proc. of LICS*, 1–10.

Bourhis, P.; Morak, M.; and Pieris, A. 2013. The impact of disjunction on query answering under guarded-based existential rules. In *Proc. of IJCAI*.

Calì, A., and Kifer, M. 2006. Containment of conjunctive object meta-queries. In *Proc. of VLDB*, 942–952.

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.* 48:115–174.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.* 14:57–83.

Calì, A.; Gottlob, G.; and Pieris, A. 2012. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.* 193:87–128.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning* 39(3):385–429.

Dantsin, E.; Eiter, T.; Georg, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Comput. Surv.* 33(3):374–425.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data exchange: Semantics and query answering. *Theor. Comput. Sci.* 336(1):89–124.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of ICLP/SLP*, 1070–1080.

Gogacz, T., and Marcinkowski, J. 2013. Converging to the chase - a tool for finite controllability. In *Proc. of LICS*, 540–549.

Gottlob, G., and Schwentick, T. 2012. Rewriting ontological queries into small nonrecursive datalog programs. In *Proc. of KR*.

Gottlob, G.; Hernich, A.; Kupke, C.; and Lukasiewicz, T. 2012a. Equality-friendly well-founded semantics and applications to description logics. In *Proc. of AAAI*, 757–764.

Gottlob, G.; Manna, M.; Morak, M.; and Pieris, A. 2012b. On the complexity of ontological reasoning under disjunctive existential rules. In *Proc. MFCS*, 1–18.

Gottlob, G.; Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2013. Query answering under probabilistic uncertainty in Datalog+/– ontologies. *Ann. Math. Artif. Intell.* 69(1):37–72.

Gottlob, G.; Kikot, S.; Kontchakov, R.; Podolskii, V.; Schwentick, T.; and Zakharyaschev, M. 2014a. The price of query rewriting in ontology-based data access. *Artif. Intell.* To appear.

Gottlob, G.; Hernich, A.; Kupke, C.; and Lukasiewicz, T. 2014b. Stable model semantics for guarded existential rules and description logics. In *Proc. of KR*.

Gottlob, G.; Manna, M.; and Pieris, A. 2013. Combining decidability paradigms for existential rules. *TPLP* 13(4-5):877–892.

Gottlob, G.; Manna, M.; and Pieris, A. 2014a. Polynomial combined rewritings for existential rules. In *Proc. of KR*.

Gottlob, G.; Manna, M.; and Pieris, A. 2014b. Querying hybrid fragments of existential rules. Forthcoming.

Gottlob, G.; Orsi, G.; and Pieris, A. 2011. Ontological queries: Rewriting and optimization. In *Proc. of ICDE*, 2–13.

Gottlob, G.; Orsi, G.; and Pieris, A. 2014. Database techniques for consistency checking of re-engineered UML class diagrams. Under review.

Gottlob, G.; Pieris, A.; and Tendera, L. 2013. Querying the guarded fragment with transitivity. In *Proc. of ICALP*, 287–298.

Gottlob, G.; Rudolph, S.; and Simkus, M. 2014. Expressiveness of guarded existential rule languages. In *Proc. of PODS*. To appear.

Grau, B. C.; Horrocks, I.; Krötzsch, M.; Kupke, C.; Magka, D.; Motik, B.; and Wang, Z. 2013. Acyclicity notions for existential rules and their application to query answering in ontologies. *J. Artif. Intell. Res.* 47:741–808.

Hernich, A.; Kupke, C.; Lukasiewicz, T.; and Gottlob, G. 2013. Well-founded semantics for extended Datalog and ontological reasoning. In *Proc. of PODS*, 225–236.

König, M.; Leclère, M.; Mugnier, M.-L.; and Thomazo, M. 2012. A sound and complete backward chaining algorithm for existential rules. In *Proc. of RR*, 122–138.

Krötzsch, M., and Rudolph, S. 2011. Extending decidable existential rules by joining acyclicity and guardedness. In *Proc. of IJCAI*.

Leone, N.; Pfeifer, G.; Faber, W.; Eiter, T.; Gottlob, G.; Perri, S.; and Scarcello, F. 2006. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Log.* 7(3):499–562.

Leone, N.; Manna, M.; Terracina, G.; and Veltri, P. 2012. Efficiently computable Datalog$^{\exists}$ programs. In *Proc. of KR*.

Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2012. Inconsistency handling in Datalog+/– ontologies. In *Proc. of ECAI*.

Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2013. Preference-based query answering in Datalog+/– ontologies. In *Proc. of IJCAI*, 1017–1023.

Marnette, B. 2009. Generalized schema-mappings: from termination to tractability. In *Proc. of PODS*, 13–22.

Orsi, G., and Pieris, A. 2011. Optimizing query answering under ontological constraints. *PVLDB* 4(11):1004–1015.

Rosati, R. 2011. On the finite controllability of conjunctive query answering in databases under open-world assumption. *J. Comput. Syst. Sci.* 77(3):572–594.

Virgilio, R. D.; Orsi, G.; Tanca, L.; and Torlone, R. 2012. Nyaya: A system supporting the uniform management of large sets of semantic data. In *Proc. of ICDE*, 1309–1312.