Querying Large Expressive Horn Ontologies

(discussion paper)

Carlo Allocca[3], Mario Alviano[1], Francesco Calimeri[1,2], Cristina Civili[3], Roberta Costabile[1], Bernardo Cuteri[1], Alessio Fiorentino[1], Davide Fuscà[1], Stefano Germano[2], Giovanni Laboccetta[2], Nicola Leone[1], Marco Manna[1], Simona Perri[1], Kristian Reale[2], Francesco Ricca[1], Pierfrancesco Veltri[2], and Jessica Zangari[1,2]

[1] Department of Mathematics and Computer Science, University of Calabria, Italy
{alviano,calimeri,r.costabile,cuteri,fiorentino,fusca, leone,manna,perri,ricca,zangari}@mat.unical.it
[2] DLVSystem L.T.D., Polo Tecnologico Unical, Italy
{calimeri,germano,laboccetta,reale,veltri}@dlvsystem.com
[3] Samsung R&D Institute, UK
{c.allocca,c.civili}@samsung.com

Abstract. In the development of the Semantic Web researchers and industry experts agree that scalability can only be obtained by reducing standard reasoning tasks to query evaluation over (deductive) databases. From a theoretical viewpoint much has been done. Conversely, from a practical point of view, only a few reasoning services have been developed, which however typically can only deal with lightweight ontologies. To fill the gap, the paper presents OWL2DLV, a modern Datalog system for evaluating SPARQL queries over very large OWL 2 knowledge bases. OWL2DLV builds on the well-known ASP system DLV by incorporating novel optimizations sensibly reducing memory consumption and a server-like behavior to support multiple-query scenarios. The high potential of OWL2DLV is outlined by the results of an experiment on data-intensive benchmarks, and confirmed by the interest of a major international industrial player, which has stimulated and partially supported the work.

Keywords: Datalog · Ontology-based query answering · DLV

1 Introduction

In large-scale Semantic Web scenarios, it is convenient to reduce standard reasoning tasks to query evaluation over (deductive) databases. From a theoretical viewpoint much has been done: in many ontological settings, the problem of evaluating a conjunctive query (CQ) over a *knowledge base* (KB) consisting of an *extensional dataset* (ABox) paired with an *ontology* (TBox) can be reduced to the evaluation of a Datalog query (i.e., a Datalog program, possibly nonrecursive and including strong constrains, paired with a union of CQs, both constructed only from the original query and the TBox) over the same ABox [14,17,22,29,33]. From a practical viewpoint the situation is not so rosy. Many Datalog reasoners, such as CLINGO [18] and DLV [26], are based on one-shot executions performing heavy operations (e.g., loading and indexing) multiple times and hence are rather unsuited. Also, only a few services with a server-like behavior, such as MASTRO [13], ONTOP [12], and RDFOX [28], have been developed, which however can only deal with lightweight TBoxes. To fill the gap, the paper presents OWL2DLV, a modern Datalog system, based on the aforementioned rewriting approach, for evaluating SPARQL CQs [32] over very large OWL 2 KBs [15].

Reasoning over OWL 2 is generally a very expensive task: fact entailment is already 2NEXPTIME-hard, while decidability of CQ answering is even an open problem. To balance expressiveness and scalability, the W3C identified three tractable profiles —OWL 2 EL, OWL 2 QL, and OWL 2 RL— exhibiting good computational properties: the evaluation of CQs over KBs falling in these fragments is in PTIME in data complexity (query and TBox are considered fixed) and in PSPACE in combined complexity (nothing is fixed) [30]. To deal with a wide variety of ontologies, OWL2DLV implements the Horn-$\mathcal{SHIQ}$ fragment of OWL 2, which enjoys good computational properties: CQs are evaluated in PTIME (resp., EXPTIME) in data (resp., combined) complexity. Moreover, it is also quite expressive: it generalizes both OWL 2 QL and OWL 2 RL, while capturing all OWL 2 EL constructs except role chain [25].

From the technical side, OWL2DLV builds on the well-known ASP system DLV [26], and in particular its most recent incarnation DLV2 [2], by incorporating a server-like modality, which is able to keep the main process alive, receive and process multiple user's requests on demand, and restore its status thanks to an embedded persistency layer. Following the approach proposed by Eiter et al. [17], a Horn-$\mathcal{SHIQ}$ TBox paired with a SPARQL query are rewritten, independently from the ABox, into an equivalent Datalog query.

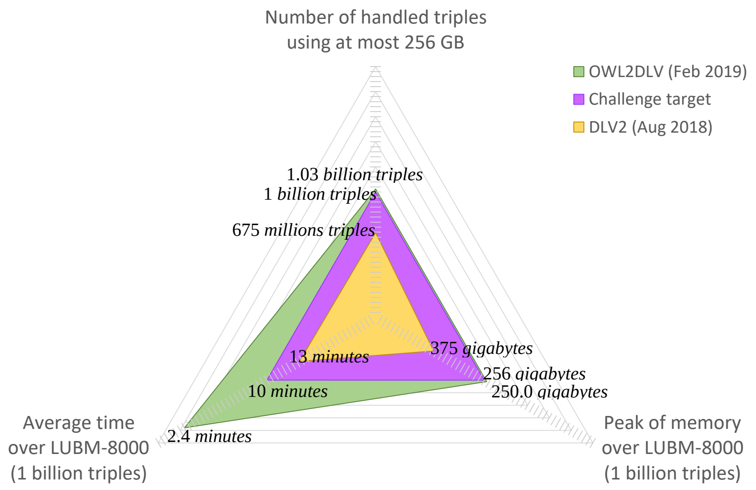

Fig. 1: OWL2DLV – performance and enhancements.

The high potential of OWL2DLV is outlined by the results of an experiment on data-intensive benchmarks, and confirmed by the direct interest of a big international industrial player, which has partially supported this work and also stimulated the evolution of the system with a major *challenge*: "deal with LUBM-8000 —the well-known LUBM [19] standard benchmark for ontological reasoning collecting about 1 billion factual assertions upon 8,000 universities— over machines equipped with 256GB RAM and with an average query evaluation time of at most 10 minutes". Eventually, not only the system was able to widely win the general challenge as reported in Figure 1; but, amazingly, the average time taken by OWL2DLV on the ten (out of fourteen) *bound* queries —i.e., queries containing at least one constant— of LUBM-8000 was eventually less than one second.

2 Background

OWL 2. In Description Logic (DL) terminology, let $\mathsf{N_I}$ (*individuals*), $\mathsf{N_C} \supset \{\top, \bot\}$ (*atomic concepts*) and $\mathsf{N_R}$ (*role names*) be pairwise disjoint discrete sets. A *role* r is either a role name s or its *inverse* s^-. A *concept* is either an atomic concept or of the form $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall r.C$, $\exists r.C$, $\geqslant nr.C$ or $\leqslant nr.C$, where C and D are concepts, r is a role, and $n \geq 1$. *General concept inclusions* (GCIs), *role inclusions* (RIs), and *transitive axioms* (TAs) are respectively of the form $C_1 \sqsubseteq C_2$, $r_1 \sqsubseteq r_2$, and $Tr(r)$, where: $\sqcup$ is disallowed in C_2, $\geqslant nr$ and $\leqslant nr$ are disallowed in C_1, and they are disallowed also in C_2 in case r is transitive. A Horn-$\mathcal{SHIQ}$ TBox is a finite set of GCIs, RIs and TAs satisfying some non-restrictive global conditions [21,23]. An instance I is a set of assertions of the form $C(a)$ and $r(a, b)$, where $C \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and $a, b \in \mathsf{N_I}$. An ABox is any finite instance. A KB $\mathcal{K}$ is a pair $(\mathcal{A}, \mathcal{T})$, where $\mathcal{A}$ is an ABox and $\mathcal{T}$ is a TBox.
SPARQL. It is the standard language in the Semantic Web for querying OWL 2 KBs [32]. As in databases, the most important class of SPARQL queries are the *conjunctive* ones, which syntactically are quite similar to SQL queries. More generally, when querying OWL 2 KBs the TBox plays the role of a fist-order theory and it has to be taken into account properly, as described next.
OBQA. By adopting the *unique name assumption*, a *model* of a KB $\mathcal{K} = (\mathcal{A}, \mathcal{T})$ is an instance $I \supseteq \mathcal{A}$ satisfying all the axioms of $\mathcal{T}$, written $I \models \mathcal{T}$ [6]. The set of all models of $\mathcal{K}$ is denoted by $\mathsf{mods}(\mathcal{K})$. To comply with the *open world assumption*, I might contain individuals that do not occur in $\mathcal{K}$. The *answers* to a query $q(\bar{x})$ over an instance I is the set $q(I) = \{\bar{a} \in \mathsf{N_I}^{|\bar{x}|} \mid I \models q(\bar{a})\}$ of $|\bar{x}|$-tuples of individuals obtained by evaluating q over I. Accordingly, the *certain answers* to q is the set $\mathsf{cert}(\mathcal{K}, q) = \bigcap_{I \in \mathsf{mods}(D,\Sigma)} q(I)$. Finally, ontology-based query answering (OBQA) is the problem of computing $\mathsf{cert}(\mathcal{K}, q)$.

3 Query Answering in OWL 2 via Datalog

As said, to perform OBQA, OWL2DLV follows the approach of Eiter et al. [17]. From an OWL 2 Horn-$\mathcal{SHIQ}$ TBox $\mathcal{T}$ and a SPARQL CQ $q(\bar{x})$, OWL2DLV runs Algorithm 1 to build a Datalog program $P_{\mathcal{T}}$ and a union of CQs $Q_{q,\mathcal{T}}(\bar{x})$ such that, for each ABox $\mathcal{A}$, the evaluation of $Q_{q,\mathcal{T}}(\bar{x})$ over $\mathcal{A} \cup P_{\mathcal{T}}$ produces the same answers as the evaluation of $q(\bar{x})$ over $\mathcal{A} \cup \mathcal{T}$.

Algorithm 1: TBox and Query Rewriting

```
Input: An OWL 2 Horn-$\mathcal{SHIQ}$ TBox $\mathcal{T}$ together with a query $q(\bar{x})$
Output: The Datalog program $P_{\mathcal{T}}$ together with the query $Q_{q,\mathcal{T}}(\bar{x})$
1. $\mathcal{T}' \leftarrow \mathsf{Normalize}(\mathcal{T})$;
2. $\mathcal{T}^* \leftarrow \mathsf{EmbedTransitivity}(\mathcal{T}')$;
3. $\Xi(\mathcal{T}^*) \leftarrow \mathsf{Saturate}(\mathcal{T}^*)$;
4. $P_{\mathcal{T}} \leftarrow \mathsf{RewriteTBox}(\Xi(\mathcal{T}^*))$;
5. $Q_{q,\mathcal{T}}(\bar{x}) \leftarrow \mathsf{RewriteQuery}(q(\bar{x}), \Xi(\mathcal{T}^*))$;
```

Consider a TBox $\mathcal{T}$. Step 1 transforms $\mathcal{T}$ into an equivalent TBox $\mathcal{T}'$ containing only "simple" axioms [21]. Step 2 transforms $\mathcal{T}'$ into $\mathcal{T}^*$ by replacing TAs with suitable GCIs including new atomic concepts. Step 3 exhaustively applies inference rules to derive new entailed axioms. The new TBox including this extra axioms is denoted by $\Xi(\mathcal{T}^*)$. Step 4, identifies the DL axioms with no existential restrictions in the right-hand side and transforms them into Datalog rules. Finally, given a CQ $q(\bar{x})$, Step 5 rewrites it into the union of CQ (UCQ) $Q_{q,\mathcal{T}}(\bar{x})$ by incorporating parts of $\Xi(\mathcal{T}^*)$.

The pair $(Q_{q,\mathcal{T}}(\bar{x}), P_{\mathcal{T}})$ returned by Algorithm 1 is further optimized by a *pruning strategy* followed by the so-called *Magic Sets rewriting* —the latter is already in use in DLV2 but it has been further improved due to the specific nature of $P_{\mathcal{T}}$. The result of this phase consists of the pair $(\mathsf{opt}(Q_{q,\mathcal{T}}(\bar{x})), \mathsf{opt}(P_{\mathcal{T}}))$.

Pruning Strategy. Pairs of GCIs of the form $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$ give rise to Datalog queries containing rules with multiple predicates having the same extensions: $C_1(X) :- C_2(X)$ and $C_2(X) :- C_1(X)$. The same happens with RIs. During the evaluation of the query, however, this can be considerably expensive. Hence, we adopt the following pruning strategy. Let $\mathcal{E} = \{E_1, ..., E_k\}$ be a set of equivalent concepts or roles. First, we remove from $P_{\mathcal{T}}$ all the rules of the form $E_i(X) :- E_j(X)$ with $1 < i \leq k$ and $1 \leq j \leq k$. Second, let $P^1_{\mathcal{T}}$ be the subset of $P_{\mathcal{T}}$ containing only rules of the form $E_1(X) :- E_j(X)$ with $2 \leq j \leq k$, for each $i \in \{2, ..., k\}$, we replace each occurrence of E_i by E_1 both in $Q_{q,\mathcal{T}}(\bar{x})$ and in each rule of $P_{\mathcal{T}}$ that does not belong to $P^1_{\mathcal{T}}$. An analogous technique is applied over RIs of the form $r \sqsubseteq s^-$ and $s^- \sqsubseteq r$ by taking into account, in this case, that the first argument of r (resp., s) maps the second one of s (resp., r).

Magic Sets Rewriting. To optimize the rewriting, OWL2DLV performs two novel steps: (1) Eliminate rules that have a magic atom with predicate `m#p#`α if $\alpha \neq$ `f`$\cdots$`f` and `m#p#f`$\cdots$`f` also occurs in the rewritten program; and (2) Remove every rule r_1 whenever it is subsumed by some other rule r_2 ($r_1 \sqsubseteq r_2$), namely there is a variable substitution mapping the head (resp., body) of r_2 to the head (resp., body) of r_1. To avoid the quadratic number of checks, OWL2DLV associates each rule with a suitable hash value of size 64 bits. Then, $r_1 \sqsubseteq r_2$ is checked only if the bit-a-bit equation $hash(r_1)$ & $hash(r_2) == hash(r_2)$ is satisfied.

4 System architecture

The OWL2DLV architecture is depicted in Figure 2. The system features four main modules: Loading, Rewriting, Query Answering, and Command Interpreter. Clients

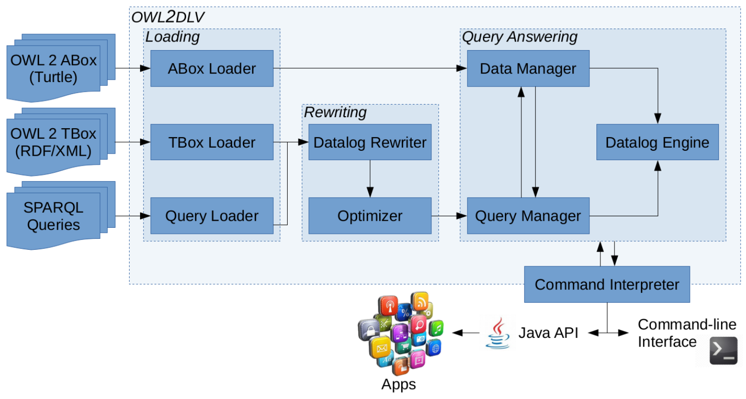

Fig. 2: System Architecture

interact with the system through the latter one. This module allows to "keep alive" the system, and execute multiple commands (e.g., loading, warmup, data updates, and query evaluations) without having to instantiate a new process for each client request. The Loading module processes an OWL 2 ABox encoded in Turtle [8] via the ABox Loader. The result of the parsing phase are Datalog-like facts stored in the OWL2DLV data structures handled by the Data Manager. Concerning the TBox, OWL2DLV supports OWL 2 Horn-$\mathcal{SHIQ}$ ontologies encoded in RDF/XML [15]. The input is parsed by the TBox Loader using the OWL API [20] and loaded in DL-like data structures. The system supports a set of SPARQL CQs via the Query Loader. Then, the Rewriting module implements Algorithm 1 via the Datalog Rewriter submodule and optimize —by applying the *pruning strategy* and the *Magic Sets rewriting*— its output via the Optimizer. The Rewriting module is in charge of evaluating Datalog queries over the parsed ABox, and producing the answers. Finally, the Query Answering module represents an extension of I-DLV [10] (the grounder of DLV2). The overall evaluation procedure is based on a bottom-up process based on a semi-naïve approach [31] empowered with optimizations working in synergy [10,11] and extended via techniques devised to manage efficiently large sizes of data.

5 Experimental Evaluation

We report the results of an experimental evaluation of OWL2DLV over LUBM [19] and DBpedia [5]. Note that, by focusing on the few ready-to-use OWL 2 reasoning services with a server-like behavior, neither MASTRO nor ONTOP nor RDFOX fully support query answering in both domains: all of them do not process some LUBM axioms. Further experiments with computationally intensive benchmarks, such as LUBM$^{\exists}$ [27] and UOBM [24], will be part of an extended version of this paper. Moreover, a comparison against MASTRO, ONTOP and RDFOX on lightweight ontologies as well as a comparison against modern Datalog-based systems like VADALOG [9] and GRAAL [7] for query existential rules [3,4] is also in our agenda.

	LUBM-8000				DBpedia			
	CQ name	*informed*	*responsive*	*dynamic*	CQ name	*informed*	*responsive*	*dynamic*
	q01	0.00	0.00	71.77	**q01**	0.18	0.32	2.51
	q02	194.48	179.65	295.74	**q02**	0.17	0.29	2.15
	q03	0.00	0.00	160.95	**q03**	0.22	0.32	2.29
	q04	0.01	0.01	379.22	**q04**	0.21	0.30	0.33
	q05	0.03	0.03	19.60	**q05**	0.20	0.29	7.52
	q06	844.65	854.35	978.46	**q06**	0.19	0.29	0.38
	q07	0.01	0.01	5.07	**q07**	0.19	0.38	0.79
	q08	0.40	0.32	0.50	**q08**	0.18	0.32	0.31
	q09	972.63	1,008.43	1,053.82				
	q10	0.00	0.01	4.66				
	q11	0.00	0.00	0.00				
	q12	0.03	0.03	0.03				
	q13	6.32	6.69	8.13				
	q14	14.64	14.50	14.12				
Max Memory		250.0	251.0	249.3		63.6	96.4	106.2
Loading		5,226.8	5,196.4	5,186.8		3,845.3	3,719.0	3,811.2
Warmup		4,144.4	3,102.3	1,303.8		226.5	1,136.3	595.5
Query Answering (all)		145.2	147.4	238.9		0.2	0.3	2.0
Query Answering (bound)		0.7	0.7	106.2		0.2	0.3	2.0

Table 1: Experimental evaluation of OWL2DLV in different scenarios. Bound queries are reported in bold. The rows "Query Answering (all)" and "Query Answering (bound)" show, respectively, the average evaluation time computed over all queries and bound queries only. Times are expressed in seconds and memory peaks in GB.

Benchmarks. LUBM is the prime choice of our industrial partner for the challenge. It describes a very-large real-world application domain encoded in OWL 2 Horn-$\mathcal{SHIQ}$ with customizable and repeatable synthetic data. The benchmark incorporates 14 SPARQL queries, 10 of which are *bound* (i.e., containing at least a constant). When rewritten together with the TBox, each LUBM query gives rise to a Datalog query consisting of about 130 rules. Data generation is carried out by the LUBM data generator tool (UBA) whose main parameter is the number of universities to consider: 8,000 in our case, for a total number of about 1 billion triples. This dataset is next referred to as LUBM-8000. Concerning DBpedia, it is a well-known KB created with the aim of sharing on the Web the multilingual knowledge collected by Wikimedia projects in a machine-readable format. For this benchmarks, we inherited a set of queries from an application conceived to query DBpedia in natural language applying the approach [16]. When rewritten together with the TBox, each DBpedia query gives rise to a Datalog query of almost 5400 rules. The latest release of the official DBpedia dataset consists of 13 billion pieces of multilingual information (RDF triples). Here, we focus on the information extracted from the English edition of Wikipedia that is composed by about half a billion triples (https://wiki.dbpedia.org/public-sparql-endpoint).

Results and Discussion. The machine used for testing is a Dell server with an Intel Xeon Gold 6140 CPU composed of 8 physical CPUs clocked at 2.30 GHz, with 297GB of RAM, and running a Linux Operating System. According to the challenge, a memory limit of 256GB has been set during all tests. Table 1 shows the results of our analysis; bound queries are reported in bold. The upper part of the table reports times taken by OWL2DLV to answer queries under the three scenarios: *informed*, *responsive*, *dynamic*; the second part shows extra statistics about peaks of memory, loading and warmup times, and average answering times computed over all queries and over bound queries only. In the *informed* scenario —where we assume that the TBox and the template queries are known in advance— we obtain the best performance. Despite the large ABox sizes, on LUBM almost all bound queries are answered in less than 0.1 seconds leading to an average time of about 0.7 seconds, while on DBpedia the average evaluation time over all queries is about 0.2 seconds. These results show the effectiveness of the Magic Sets technique to dramatically reduce the part of the dataset needed to answer the query, together with all other optimization strategies discussed through the paper. In the *responsive* scenario —where only the TBox is known in advance— the system performance is comparable with the one obtained in the *informed* scenario, although in general the evaluation time is a bit higher. These results confirm that the warmup policy of this setting has a positive impact on the system performance although queries are not known. Finally, in the *dynamic* scenario —where nothing is known— the parsimonious indexing policy is adopted since 256GB are not enough to use the aggressive one. Uniformly, the same policy is also used for DBpedia although not expressly needed. This produces a general gain in the warmup phase later unavoidably paid during the query evaluation due to some missing index that has to be computed on-the-fly. For further details we refer to [1].

Acknowledgments. This work has been partially supported by Samsung under project "Enhancing the DLV system for large-scale ontology reasoning", by MISE under project "S2BDW" (F/050389/01-03/X32) – "Horizon2020" PON I&C2014-20 and by Regione Calabria under project "DLV LargeScale" (CUP J28C17000220006) – POR Calabria 2014-20.

References

1. Allocca, C., Calimeri, F., Civili, C., Costabile, R., Cuteri, B., Fiorentino, A., Fuscà, D., Germano, S., Laboccetta, G., Manna, M., Perri, S., Reale, K., Ricca, F., Veltri, P., Zangari, J.: Large-scale reasoning on expressive horn ontologies. In: Datalog 2.0. CEUR-WS.org (2019)
2. Alviano, M., Calimeri, F., Dodaro, C., Fuscà, D., Leone, N., Perri, S., Ricca, F., Veltri, P., Zangari, J.: The ASP system DLV2. In: LPNMR (2017)
3. Amendola, G., Leone, N., Manna, M.: Finite model reasoning over existential rules. TPLP **17**(5-6), 726–743 (2017)
4. Amendola, G., Leone, N., Manna, M., Veltri, P.: Enhancing existential rules by closed-world variables. In: IJCAI. pp. 1676–1682 (2018)
5. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: Dbpedia: A nucleus for a web of open data. In: ISWC. pp. 722–735. LNCS (2007)
6. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. C.U.P. (2003)
7. Baget, J., Leclère, M., Mugnier, M., Rocher, S., Sipieter, C.: Graal: A toolkit for query answering with existential rules. In: RuleML. pp. 328–344 (2015)
8. Beckett, D., Berners-Lee, T., Prud'hommeaux, E., Carothers, G., Machina, L.: Rdf 1.1 turtle – terse rdf triple language. W3C Recommendation (2014)
9. Bellomarini, L., Sallinger, E., Gottlob, G.: The vadalog system: Datalog-based reasoning for knowledge graphs. PVLDB **11**(9), 975–987 (2018)

10. Calimeri, F., Fuscà, D., Perri, S., Zangari, J.: I-DLV: the new intelligent grounder of DLV. Intelligenza Artificiale **11**(1), 5–20 (2017)
11. Calimeri, F., Perri, S., Zangari, J.: Optimizing answer set computation via heuristic-based decomposition. TPLP p. 126 (2019)
12. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: Ontop: Answering SPARQL queries over relational databases. Semantic Web **8**(3), 471–487 (2017)
13. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The MASTRO system for ontology-based data access. Semantic Web **2**(1), 43–53 (2011)
14. Carral, D., Dragoste, I., Krötzsch, M.: The combined approach to query answering in horn-alchoiq. In: KR. pp. 339–348. AAAI Press (2018)
15. Carroll, J., Herman, I., Patel-Schneider, P.F.: Owl 2 web ontology language rdf-based semantics (second edition) (2012)
16. Cuteri, B., Reale, K., Ricca, F.: A logic-based question answering system for cultural heritage. In: JELIA. LNCS, Springer (to appear) (2019)
17. Eiter, T., Ortiz, M., Simkus, M., Tran, T., Xiao, G.: Query rewriting for horn-shiq plus rules. In: AAAI (2012)
18. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with clingo 5. In: ICLP TCs. pp. 2:1–2:15 (2016)
19. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Semant. **3**(2-3), 158–182 (2005)
20. Horridge, M., Bechhofer, S.: The OWL API: A java API for OWL ontologies. Semantic Web **2**(1), 11–21 (2011)
21. Kazakov, Y.: Consequence-driven reasoning for horn SHIQ ontologies. In: IJCAI. pp. 2040–2045 (2009)
22. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: IJCAI (2011)
23. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexities of horn description logics. ACM Trans. Comput. Log. **14**(1), 2:1–2:36 (2013)
24. Ledvinka, M., Kremen, P.: Object-uobm: An ontological benchmark for object-oriented access. In: KESW. vol. 518, pp. 132–146. Springer (2015)
25. Leone, N.: The AI system DLV: ontologies, reasoning, and more. In: IC3K. pp. 5–16 (2018)
26. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. TOCL (2006)
27. Lutz, C., Seylan, I., Toman, D., Wolter, F.: The combined approach to OBDA: taming role hierarchies using filters. In: ISWC. pp. 314–330 (2013)
28. Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., Banerjee, J.: Rdfox: A highly-scalable RDF store. In: ISWC. vol. 9367, pp. 3–20 (2015)
29. Stefanoni, G., Motik, B., Horrocks, I.: Small datalog query rewritings for EL. In: DL. CEUR Workshop Proceedings, vol. 846 (2012)
30. Stefanoni, G., Motik, B., Krötzsch, M., Rudolph, S.: The complexity of answering conjunctive and navigational queries over OWL 2 EL knowledge bases. J. Artif. Intell. Res. **51**, 645–705 (2014)
31. Ullman, J.D.: Principles of Database and Knowledge-Base Systems, Volume I. Computer Science Press (1988)
32. W3C: SPARQL 1.1 entailment regimes. https://www.w3.org/TR/sparql11-entailment/ (2013)
33. Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., Zakharyaschev, M.: Ontology-based data access: A survey. In: IJCAI (2018)