Ontology-based Query Answering with PAGOdA

Yujiao Zhou, Yavor Nenov, Bernardo Cuenca Grau, and Ian Horrocks

Department of Computer Science, University of Oxford, UK

1 Introduction

We describe PAGOdA: a highly optimised 'pay-as-you-go' reasoning system that supports conjunctive query (CQ) answering with respect to an arbitrary OWL 2 ontology and an RDF dataset. PAGOdA uses a novel hybrid approach to query answering that combines a datalog reasoner (currently RDFox [10]) with a fully-fledged OWL 2 reasoner (currently HermiT [5]) to provide scalable performance while still guaranteeing sound and complete answers.[1] PAGOdA delegates the bulk of the computational workload to the datalog reasoner, with the extent to which the fully-fledged reasoner is needed depending on interactions between the ontology, the dataset and the query. Thus, even when using a very expressive ontology, queries can often be fully answered using only the datalog reasoner; and even when the fully-fledged reasoner is required, PAGOdA employs a range of optimisations to reduce the number and size of the relevant reasoning problems.

This approach has proved to be very effective in practice: in our tests of more than 2,000 queries over five ontologies, none of which is contained within any of the OWL profiles, more than 99% of queries were fully answered without resorting to the fully-fledged reasoner. Moreover, even when the fully-fledged reasoner was used, the above mentioned optimisations were highly effective: the size of the dataset was typically reduced by an order magnitude, and often by several orders of magnitude, and it seldom required more than a single test to resolve the status of all potential answer tuples. Taken together, our experiments demonstrate that PAGOdA can provide an efficient CQ answering service in real-world scenarios requiring both expressive ontologies and datasets containing hundreds of millions of facts.

The basic approach in PAGOdA has been described in [13, 14, 12], and full details about the algorithms currently implemented can be found in an accompanying technical report.[2] In this paper, we provide an overview of the system and highlight some of the results from our extensive evaluation.

[1] In practice we are limited by the capabilities of OWL 2 reasoners, which typically restrict the structure of the ontology and/or query in order to ensure decidability (which is open for CQ answering over unrestricted OWL 2 ontologies).

[2] `http://www.cs.ox.ac.uk/isg/tools/PAGOdA/pagoda-tr.pdf`

2 The PAGOdA System

PAGOdA is written in Java and it is available under an academic license.[3] As well as RDFox and HermiT, PAGOdA also exploits the combined approach for CQ answering in $\mathcal{ELHO}^r_\bot$ implemented in KARMA.[4] The architecture of PAGOdA is depicted in Figure 1.

PAGOdA accepts as input arbitrary OWL 2 DL ontologies, datasets in Turtle format and CQs in SPARQL. Queries can be interpreted under ground or certain answer semantics. In the former case, PAGOdA is sound and complete. In the latter case, however, PAGOdA is ultimately limited by the capabilities of HermiT, which can only check entailment of ground or DL concept queries; hence, PAGOdA can guarantee completeness only if the lower and upper bounds match, or if the query can be transformed into a DL concept query via rolling-up.[5] Otherwise, PAGOdA returns a sound (but possibly incomplete) set of answers, along with a bound on the incompleteness of the computed answer set.

PAGOdA uses four instances of RDFox: one for the lower bound, one for each upper bound materialisation (c-chase and c-chase^f), and one for subset extraction. Furthermore, it uses two instances of HermiT (one in each of the summary filter and dependency graph components).

The process of fully answering a query can be divided into several steps. Here, we distinguish between query independent steps and query dependent ones. As we can see in Figure 1, the 'loading ontology' and 'materialisation' steps are query independent. Therefore, both of them are counted as *pre-processing* steps. 'Computing query bounds', 'extracting subset' and 'full reasoning' are query dependent, and are called *query processing* steps.

Loading ontology and data. PAGOdA uses the OWL API to parse the input ontology $\mathcal{O}$. The dataset $\mathcal{D}$ is given separately in Turtle format. The normaliser then transforms the ontology into a set of rules corresponding to the axioms in $\mathcal{O}$. PAGOdA's normaliser is an extension of HermiT's clausification component [5], which transforms axioms into so-called DL-clauses [11]. The dataset $\mathcal{D}$ is loaded directly into (the four instances of) RDFox.

After normalisation, the ontology is checked to determine if it is inside OWL 2 RL (resp. $\mathcal{ELHO}^r_\bot$); if so, then RDFox (resp. KARMA) is already sound and complete, and PAGOdA simply processes $\mathcal{O}$, $\mathcal{D}$ and subsequent queries using the relevant component. Otherwise, PAGOdA uses a variant of *shifting*—a polynomial program transformation commonly used in Answer Set Programming [4]—to enrich the deterministic part of the ontology with some additional information from disjunctive rules, resulting in a rule set Σ.

Materialisation. There are three components involved in this step, namely lower bound, c-chase and c-chase^f. Each of these takes as input Σ and $\mathcal{D}$, and each computes a materialisation (shown in Figure 1 as ellipses). The lower bound

[3] `http://www.cs.ox.ac.uk/isg/tools/PAGOdA/`

[4] `http://www.cs.ox.ac.uk/isg/tools/KARMA/`.

[5] PAGOdA implements an extension of the well-known rolling-up technique.

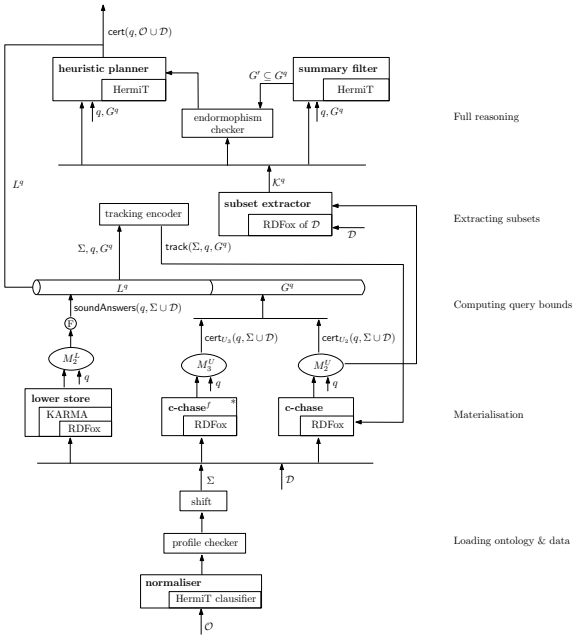


Fig. 1: The architecture of PAGOdA

component first uses RDFox to compute a materialisation of $\mathcal{D}$ using the datalog subset of Σ; it then uses the materialised dataset as input to KARMA, which computes the materialisation M_2^L using the $\mathcal{ELHO}_\bot^r$ subset of Σ. The c-chase and c-chase^f components compute the M_2^U and M_3^U upper bound materialisations using chase-like procedures [1]. The former (M_2^U) is computed by over-approximating Σ into datalog; this involves, roughly speaking, transforming disjunctions into conjunctions, and replacing existentially quantified variables with fresh constants [12]. However, PAGOdA optimises the treatment of 'Skolemised' existential rules by not applying them if the existential restriction is already satisfied in the data. In M_3^U, PAGOdA further optimises the treatment of disjunctions by selecting a single disjunct in the head of disjunctive rules, using a heuristic choice function that tries to select disjuncts that will not (eventually) lead to a contradiction.

If $\bot$ is derived while computing M_2^L, then the input ontology and dataset is unsatisfiable, and PAGOdA simply reports this and terminates. If $\bot$ is derived while computing M_3^U, then the computation is aborted and M_3^U is no longer used. If $\bot$ is derived while computing M_2^U, then PAGOdA checks the satisfiability of $\Sigma \cup \mathcal{D}$ (using the optimised query answering procedure described below). If $\Sigma \cup \mathcal{D}$ is unsatisfiable, then PAGOdA reports this and terminates; otherwise the input ontology and dataset is satisfiable, and PAGOdA is able to answer queries.

Computing query bounds. Given a query q, PAGOdA uses the M_2^L lower bound materialisation to compute the lower bound answer L^q, exploiting the filtration procedure in KARMA to eliminate spurious answer tuples (shown as

	♯axioms	♯rules	♯∃-rules	♯∨-rules	♯facts
LUBM(n)	93	133	15	0	$n \times 10^5$
UOBM(n)	186	234	23	6	$2.6n \times 10^5$
ChEMBL	2,593	2,960	426	73	2.9×10^8
Reactome	559	575	13	23	1.2×10^7
Uniprot	442	459	20	43	1.2×10^8

Table 1: Test data. Columns from left to right indicate the number of DL axioms, the number of rules after normalisation, the number of rules containing $\exists$, the number of rules containing $\vee$ and the number of facts in each dataset.

a circle with an 'F' in it in Figure 1). If $\bot$ was not derived when computing the M_3^U materialisation, then the upper bound answer U^q is the intersection of the query answers w.r.t. M_3^U and M_2^U; otherwise U^q is computed using only M_2^U.

Extracting subsets. If $L^q = U^q$, then PAGOdA simply returns L^q; otherwise it must determine the status of the tuples in the 'gap' $G^q = U^q \setminus L^q$. To do this, PAGOdA extracts subsets of Σ and $\mathcal{D}$ that are sufficient to check the entailment of each such tuple. First, the tracking encoder component is used to compute a datalog program that tracks rule applications that led to the derivation of the tuples in G^q. This program is then added to the rules and data in the c-chase component, and RDFox is used to extend the c-chase materialisation accordingly. The freshly derived facts (over the tracking predicates introduced by the tracking encoder) are then passed to the subset extractor component, which identifies the relevant facts and rules in Σ and $\mathcal{D}$.

Full reasoning. PAGOdA uses HermiT to verify answers in G^q. As HermiT only accepts queries given either as facts or DL concepts, we have implemented an extension of the rolling-up technique to internalise CQs as ontology axioms [7]. In the summary filter component, PAGOdA uses summarisation techniques inspired by the SHER system to quickly identify spurious gap tuples [2, 3]. The remaining gap answers $G' \subseteq G^q$ are then passed to the endormorphism checker, which exploits a greedy algorithm to compute a (incomplete) dependency graph between answers in G'. This graph is used by the heuristic planner to optimise the order in which the answers in G' are checked using HermiT. Finally, verified answers from G' are combined with the lower bound L^q.

3 Evaluation

We have evaluated PAGOdA on a range of realistic and benchmark ontologies, datasets and queries. Experiments were conducted on a 32 core 2.60GHz Intel Xeon E5-2670 with 250GB of RAM, and running Fedora 20.[6] Further details, including detailed comparison with other systems, additional experiments, and results on other ontologies are given in our technical report.

Table 1 summarises our test data. LUBM and UOBM are widely-used benchmarks [6, 9]. We have tested 10 additional queries for which datalog lower-bound

[6] Test data available at `http://www.cs.ox.ac.uk/isg/tools/PAGOdA/2015/jair/`

answers are not guaranteed to be complete (as is the case for the standard queries). ChEMBL, Reactome, and Uniprot are ontologies that are available from the European Bioinformatics Institute (EBI) linked data platform.[7] The ontologies are complex and the data is both realistic and large-scale. We computed subsets of the data using a sampling algorithm based on random walks [8]. We tested both realistic queries and all atomic queries.

We tested the scalability of PAGOdA on LUBM, UOBM and the ontologies from the EBI platform. For LUBM we used datasets of increasing size with a step of $n = 100$. For UOBM we also used increasingly large datasets with step $n = 100$ and we also considered a smaller step of $n = 5$ for hard queries. Finally, in the case of EBI's datasets, we computed subsets of the data of increasing sizes from 1% of the original dataset up to 100% in steps of 10%. For each test ontology we measured *pre-processing time* and *query processing time* as described in Section 2. We organise the test queries into three groups: **G1**: queries for which the lower and upper bounds coincide; **G2**: queries with a non-empty gap, but for which summarisation is able to filter out all remaining candidate answers; and **G3**: queries where the fully-fledged reasoner is called over an ontology subset on at least one of the test datasets. A timeout of 2.5h was set for each individual query and 5h for all queries.

Our results are summarised in Figure 2. For each ontology, we plot time against the size of the input dataset, and for query processing we distinguish different groups of queries as discussed above. PAGOdA behaves relatively uniformly for queries in **G1** and **G2**, so we plot only the average time per query for these groups. In contrast, PAGOdA's behaviour for queries in **G3** is quite variable, so we plot the time for each individual query.

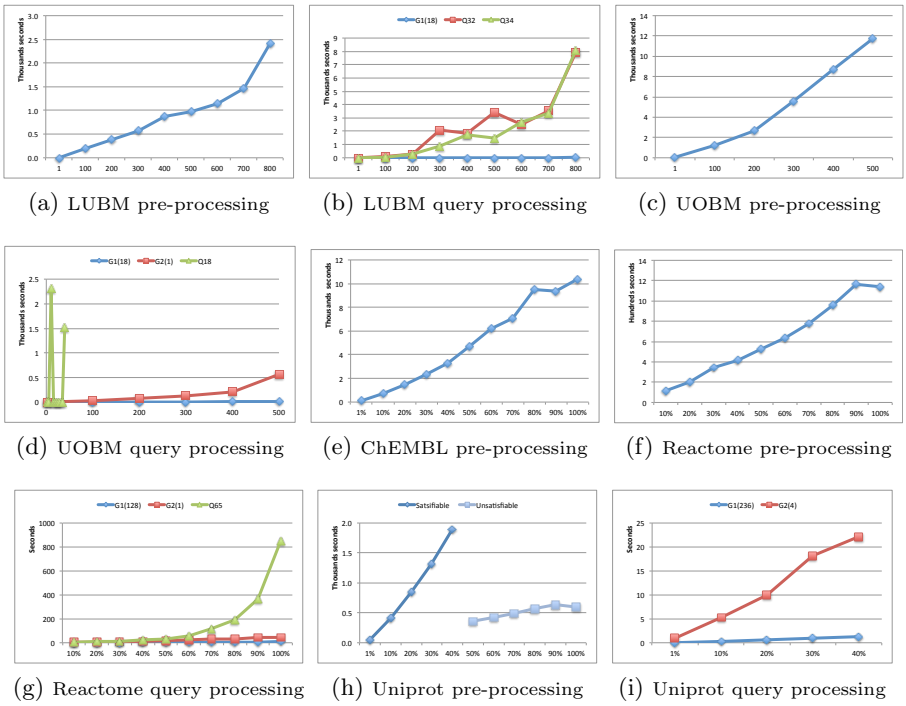
LUBM(n) All LUBM queries belongs to either **G1** or **G3** with the latter group containing two queries. The average query processing time for queries in **G1** never exceeds 13s; for the two queries in **G3** (Q32 and Q34), this reaches $8,000$s for LUBM(800), most of which is accounted for by HermiT.

UOBM(n) As with LUBM, most test queries were contained in **G1**, and their processing times never exceeded 8 seconds. We found one query in **G2**, and PAGOdA took 569s to answer this query for UOBM(500). UOBM's randomised data generation led to the highly variable behaviour of Q18: it was in **G3** for UOBM(1) UOBM(10) and UOBM(50), causing PAGOdA to time out in the last case; it was in **G2** for UOBM(40); and it was in **G1** in all other cases.

ChEMBL All test queries were contained in **G1**, and average processing time was less than 0.5s in all cases. Thus, we have not depicted the scalability of query processing for ChEMBL in Figure 2.

Reactome Groups **G2** and **G3** each contained one query, with all the remaining queries belonging to **G1**. Query processing time for queries in **G1** never exceeded 10 seconds; for **G2** processing time appeared to grow linearly in the size of datasets, and average time never exceeded 10 seconds; the **G3** query (Q65) is

[7] http://www.ebi.ac.uk/rdf/platform



(a) LUBM pre-processing (b) LUBM query processing (c) UOBM pre-processing

(d) UOBM query processing (e) ChEMBL pre-processing (f) Reactome pre-processing

(g) Reactome query processing (h) Uniprot pre-processing (i) Uniprot query processing

Fig. 2: Scalability tests

much more challenging, but it could still be answered in less than 900 seconds, even for the largest dataset.

Uniprot In contrast to the other cases, Uniprot as a whole is unsatisfiable; however, our sampling technique can produce a satisfiable subset up to 40%. For larger subsets, pre-processing times drop abruptly as unsatisfiability can be efficiently detected in the lower bound. Query processing times were only considered for satisfiable samples. There were no queries in **G3**, and only four in **G2**, all of which were efficiently handled.

4 Discussion

We have described PAGOdA: a highly scalable and robust query answering system for OWL 2 DL ontologies. Our experiments using the realistic ontologies and datasets in the EBI linked data platform have shown that PAGOdA is capable of fully answering queries over highly complex and expressive ontologies and realistic datasets containing hundreds of millions of facts—something that is far beyond the capabilities of state-of-the-art ontology reasoners.

Acknowledgements. This work has been supported by the Royal Society under a Royal Society Research Fellowship, by the EPSRC projects MaSI^3, Score! and DBOnto, and by the EU FP7 project Optique.

References

1. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. Journal of Artificial Intelligence Research 48, 115–174 (2013)
2. Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Ma, L.: Scalable semantic retrieval through summarization and refinement. In: AAAI 2007, Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada. pp. 299–304. AAAI Press (2007)
3. Dolby, J., Fokoue, A., Kalyanpur, A., Schonberg, E., Srinivas, K.: Scalable highly expressive reasoner (SHER). Journal of Web Semantics 7(4), 357–361 (2009)
4. Eiter, T., Fink, M., Tompits, H., Woltran, S.: Simplifying logic programs under uniform and strong equivalence. In: LPNMR 2004, Proceedings of Logic Programming and Nonmonotonic Reasoning - 7th International Conference, Fort Lauderdale, FL, USA, January 6-8, 2004, Proceedings. pp. 87–99 (2004)
5. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: An OWL 2 reasoner. Journal of Automated Reasoning 53(3), 245–269 (2014)
6. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics 3(2-3), 158–182 (2005)
7. Horrocks, I., Tessaris, S.: A conjunctive query language for description logic aboxes. In: AAAI/IAAI 2000, Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA. pp. 399–404. AAAI Press / The MIT Press (2000)
8. Leskovec, J., Faloutsos, C.: Sampling from large graphs. In: KDD 2006, Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006. pp. 631–636 (2006)
9. Ma, L., Yang, Y., Qiu, Z., Xie, G.T., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: ESWC 2006, The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, Budva, Montenegro, June 11-14, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4011, pp. 125–139. Springer (2006)
10. Motik, B., Nenov, Y., Piro, R., Horrocks, I., Olteanu, D.: Parallel materialisation of datalog programs in centralised, main-memory RDF systems. In: AAAI 2014, Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada. pp. 129–137. AAAI Press (2014)
11. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. Journal of Artificial Intelligence Research 36, 165–228 (2009)
12. Zhou, Y., Nenov, Y., Cuenca Grau, B., Horrocks, I.: Pay-as-you-go OWL query answering using a triple store. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (2014)
13. Zhou, Y., Nenov, Y., Grau, B.C., Horrocks, I.: Complete query answering over Horn ontologies using a triple store. In: The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I. Lecture Notes in Computer Science, vol. 8218, pp. 720–736. Springer (2013)
14. Zhou, Y., Nenov, Y., Grau, B.C., Horrocks, I.: Pay-as-you-go ontology query answering using a datalog reasoner. In: Informal Proceedings of the 27th International Workshop on Description Logics, Vienna, Austria, July 17-20, 2014. CEUR Workshop Proceedings, vol. 1193, pp. 352–364. CEUR-WS.org (2014)