FRaQuE: A Framework for Rapid Query Processing Evaluation

Jean-Rémi Bourguet and Luca Pulina

POLCOMING, Università di Sassari, Italy
Viale Mancini 5 – 07100, Sassari – Italy
boremi@uniss.it - lpulina@uniss.it

Abstract. In this paper we present FRaQuE (**F**ramework for **Ra**pid **Qu**ery Processing **E**valuation). The main purpose of FRaQuE is to offer to a non-expert user a "push-button" solution aimed to help her to evaluate query processors for Ontology Based Data Access, focusing only on input and output data, without take into account both theoretical and technical issues.

1 Introduction

The choice of W3C to make ontologies the main tool to attach semantic information to web contents, and, consequently, the potential applications in the Semantic Web [2], has attracted a lot of interest inside the Automated Reasoning community, particularly in the past decade. It is well-established that reasoning with ontologies is one of the core task of research in description logics – see, e.g., [1] – and it is also witnessed by the large amount of reasoners currently available[1].

One of the reasoning tasks that can be accomplished by reasoning tools is query answering. In particular, in order to match the competing requirements of KR&R-style data handling with DB-style data size, research on ontology-based data access (OBDA) emerged at the crossroads of the two fields. According to [7], the keyword OBDA characterizes scenarios where access to data is mediated by an ontology, and data is either physically stored in a traditional DB, or comes in sizes which are typical of DB applications anyway. To this purpose, there are actually several OWL reasoners with the ability to support the SPARQL query language [18] – the W3C standard for querying semantic-enabled data stores.

Given the wide range of possible practical applications in which OBDA can be used, e.g., Decision Support Systems [8, 5, 3], practitioners aimed to leverage OBDA in their applications have to answer the question "Which query processor should I use?". In order to answer to this question, recently several events and projects have been implemented, e.g., the Joint Workshop on Scalable and High-Performance Semantic Web Systems [11] and the SEALS project `http://www.seals-project.eu`. More, the OWL Reasoner Evaluation

[1] See, e.g., `http://www.w3.org/2007/OWL/wiki/Implementations` or `http://www.cs.man.ac.uk/~sattler/reasoners.html` for a list

workshops organizes since 2012 a competitive event, in the same spirit of other Automated Reasoning communities, e.g., the CADE ATP System Competition (CASC) [24] for theorem provers in first order logic, the QBF Evaluation [17] for quantified Boolean formulas solvers, and the ASP Competition [6] for Answer Set Programming solvers. Also if in such kind of events reasoners are evaluated using transparent and fair methods, in a practical application context a practitioner could be interested to understand the current state of the art related to a particular problem or another for which data could not be available to the research community. This can lead a non-expert user to deal with several issues, both technical and theoretical.

In this paper we present FRaQuE (**F**ramework for **Ra**pid **Qu**ery Processing **E**valuation), a framework aimed to offer to a non-expert user the possibility to evaluate query processors for OBDA.The main purpose of FRaQuE is to offer to the user a "push-button" solution aimed to help the user to answer to the question above, focusing only on input data and queries at the user execution stage, and showing data in order to evaluate both correctness and performance at the user validation stage. Currently we include in FRaQuE research prototypes that can be considered active OBDA projects as soon as systems like Pellet [23], a full-fledged commercial description logic reasoner and ARQ [21], the built-in query processor of the Jena library. An important element in the selection is the ability to support the SPARQL query language [18]. However, the FRaQuE architecture is modular, allowing the extension to other reasoners in an easy way, as we will describe in Section 2, where we will detail both design and implementation of FRaQuE. About the rest of the paper, in Section 3 we discuss about some open points coming from the usage of the query processors, and we conclude in Section 4 with some final remarks.

2 Design and implementation of FRaQuE

Figure 1 presents the architecture of FRaQuE[2]. Because of, given a knowledge base $\mathcal{K}$ and a query α, the goal of FRaQuE is to run different systems on the task of query answering, we also refer to the OBDA systems as *query processors*. Looking at the figure, we can see that FRaQuE is composed of the four modules described in the following.

`INTERFACE` manages both the input received by the user and the output of the whole system. It also dispatches the input data to both `QUERY MANAGER` and `ONTOLOGY MANAGER`, as denoted by the outgoing arrows. In particular, `INTERFACE` collects (*i*) the name of the query processor to fire; (*ii*) the TBox file name in RDF/XML or OWL/XML format; (*iii*) the ABox file name in RDF/XML or OWL/XML format; and (*iv*) a text file containing the query in SPARQL 1.0. Finally, `INTERFACE` manages the output received from `REASONER MANAGER`, in order to present it to the user. Actually, FRaQuE

[2] FRaQuE is available for download at `http://sites.google.com/site/ore2013fraque`.

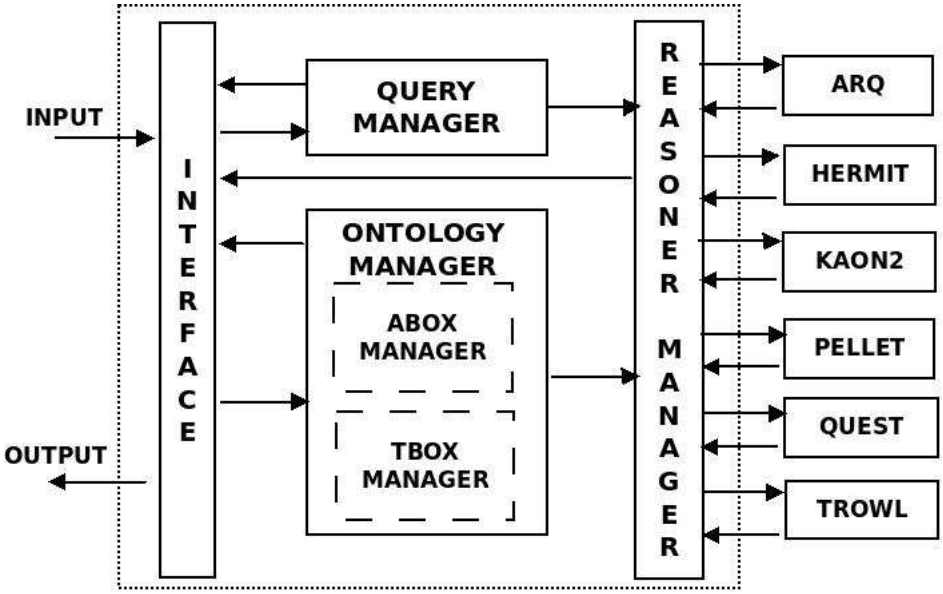

Fig. 1: The architecture of FRaQuE. The dotted box denotes the whole system and, inside it, each solid box represents its modules, while each dashed box represents a sub-module. Arrows denote functional connections between modules.

outputs the ontology loading CPU time, the query answering CPU time, and a text file containing the query result.

`QUERY MANAGER` is devoted to process the query input file received by `INTERFACE`. It checks the compliance of the query with the SPARQL 1.0 syntax, and, considering the query processor passed by `INTERFACE`, it applies syntactic modification to the input query file or returns to `INTERFACE` an error message if the input query is not supported by the selected query processor (see Section 3 for details).

`ONTOLOGY MANAGER` is devoted to manage both TBox and ABox input file, by means of sub-modules `TBOX MANAGER` and `ABOX MANAGER`, respectively.

`REASONER MANAGER` manages the interaction with the reasoners. It receives from `INTERFACE` information about the engine to fire, while it receives from `ONTOLOGY MANAGER` and `QUERY MANAGER` information about the ontology file and the query to process, respectively. At the end of the query processing, `REASONER MANAGER` returns to `INTERFACE` the result.

Concerning the modules described above, we can consider `REASONER MANAGER` as the core of FRaQuE, because it interacts with different query processors in a transparent way to the user. In particular, SPARQL expression semantics can change according to different *entailment regimes*. In our case, there are two entailment regimes which are relevant, namely the *RDFS entailment* and

the *OWL-DL entailment*. Under the second regime, we focus on the notion of entailment for the semantics of OWL 2 QL. In particular, the OWL 2 QL profile is described in the official W3C's recommendation as "*[the sub-language of OWL 2] aimed at applications that use very large volumes of instance data, and where query answering is the most important reasoning task.*". Given our intended applications, we consider knowledge bases encoded using OWL 2 QL.

In details, the query processors actually included in FRaQuE are the following.

- ARQ [21] (version 2.9.4) is the built-in query processor of the Jena library. It processes queries according to the RDFS regime.
- HermiT [22] (version 1.3.6) is a DL reasoner based on hypertableau calculus [15]. It can be used to answer sparql1.0 queries by means of the sparql1.0 wrapper owl-bgp [10, 13]. For the sake of simplicity, in the following we will mention the composition between the reasoner and the wrapper simply as "HermiT".
- kaon2 [12, 14] (version 2008-06-29) implements reasoning algorithms aimed to reduce a knowledge base to a disjunctive datalog program, allowing the usage of deductive database techniques. So, with respect to other DL reasoners like HermiT, Pellet and TrOWL, it does not implement a tableau-like calculus. Queries can be formulated using a specific subset of sparql1.0 syntax – see `http://kaon2.semanticweb.org/` for details.
- Pellet [23] (version 2.3.0) is a description logic reasoner accepting sparql1.0 queries. As such, it supports OWL-DL regime and it could be used to perform logically-aware queries also on full-fledged OWL 2 knowledge bases.
- Quest [20] (version 1.7) is a reasoner that supports the OWL-DL entailment regime restricted to OWL 2 QL. Quest converts queries over the knowledge base into equivalent SQL queries over an internal relational database. In particular, Quest uses h2 (version 1.3)[3] and while the schema used internally to store triples is similar to the standard $\langle S, P, O \rangle$ schema, it is optimized to generate very small SQL queries even if there are big hierarchies in the ontology, as described in [19].
- TrOWL [25] (version 1.1) is an infrastructure aimed to reasoning, and querying OWL 2 ontologies by means of several techniques, e.g., quality guaranteed approximations and forgetting. In general, considering OWL 2 ontologies, TrOWL could not give complete answers, but it should not be the case considering OWL 2 QL ontologies, as in our case (see [25] for details). In FRaQuE we include TrOWL with the Jena library.

Finally, in Figure 2, we present the class diagram of FRaQuE. In the diagram, we denote Java classes with boxes, "part of" relationship with hollow diamond shape arrows, while the inheritance relationship is denoted using ordinary arrows. The upper part of boxes holds the name of the class, the middle part contains the attributes and the bottom part contains methods, "+", "-" and "#" are respectively public, private and protected attributes or methods.

[3] H2 Database Engine `http://www.h2database.com/html/main.html`.

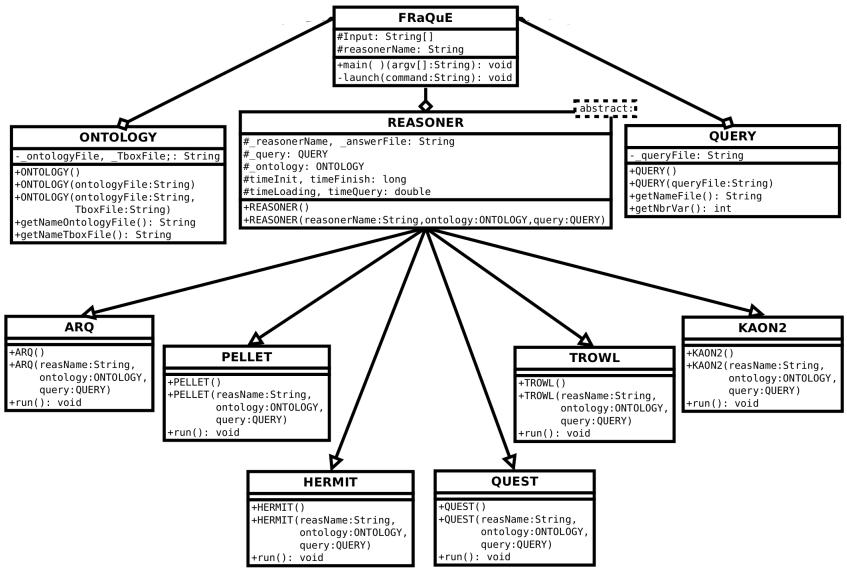

Fig. 2: Class diagram of FRaQuE.

3 Discussion

The main purpose of FRaQuE is to offer to a non-expert user a simple platform to evaluate both correctness and reasoning times of OBDA query processors. With this aim, about the evaluation of correctness of the answer, a text file containing the answer to the input query is produced by FRaQuE at the end of query processing – as mentioned in the description of the `INTERFACE` module. Concerning the reasoning time, we consider values that could be easily evaluated by a non-expert user aimed to roughly compare the CPU time needed to answer a query, avoiding technical details concerning different strategies implemented in a reasoner.

Considering query processors currently included in FRaQuE, we list in the following some technical issues. We report that queries containing some keywords, e.g., `FILTER` and `OPTIONAL`, are actually not supported by both KAON2 and Quest. About the query processors mentioned above, we also report that they do not allow the usage of variables after a `rdf:type` predicate. Some other features in the SPARQL syntax can lead to a failure during the query loading. In Quest, comment lines (starting with #) have to be removed, while in KAON2

absence of a white space between an object and the final dot is not allowed, while it is allowed between object and semi-colon[4].

The modular architecture of FRaQuE allows the user to easily extend the pool of query processors. Looking at Figure 2, we can see that all query processors currently available in FRaQuE are wrapped in a Java class derived from the abstract class REASONER. Such abstract class provides a common interface to manage input (ontology and query) and output (the query answer) files. Once implemented the derived class related to a new query processor – see the source code available at `http://sites.google.com/site/ore2013fraque` for an example – it is sufficient to update the code of FRaQuE* files.

Finally, we report that a preliminary version of FRaQuE has been used for the experimental evaluation in [4].

4 Conclusions and Future Works

In this paper, we presented the design and the implementation of FRaQuE. Our modular framework can allow to a non-expert user a rapid evaluation of the state of the art concerning query processors for OBDA.

Currently, we are working to extend FRaQuE in several directions. Firstly, we are extending the architecture in order to integrate query processors using rewriting-based techniques, e.g., CLIPPER [9] and REQUIEM [16]. Secondly, we are considering the usage of SPARQL 1.1 as input query format. This is mainly motivated by the fact that our tool aims to simplify practitioner's work, and the usage of operators like COUNT, MIN, MAX, or SUM could simplify the query formulation. Actually, ARQ is the only system supporting SPARQL 1.1 natively, so we are working on `QUERY MANAGER` in order to add a translator able to convert the input query to SPARQL 1.0 and use some JAVA code to replicate the behaviour of the operators above. Finally, we are designing a GUI version of `INTERFACE`.

Acknowledgments The authors are grateful to the reviewers for their valuable comments and suggestions for improving the paper. The authors would like to thank Giuseppe Cicala and Armando Tacchella for fruitful discussion, and Mariano Rodriguez-Muro for his help in using QUEST.

This work is supported by Regione Autonoma della Sardegna e Autorità Portuale di Cagliari con L.R. 7/2007, Tender 16 2011, CRP-49656 "Metodi innovativi per il supporto alle decisioni riguardanti lottimizzazione delle attività in un terminal container"

[4] Notice that the usage of white spaces aiming to separate two terminals is a W3C recommendation [18].

References

1. F. Baader. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
2. Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.
3. Eva Blomqvist. The use of semantic web technologies for decision support – a survey. *Semantic Web*, 2012.
4. Jean-Remi Bourguet, Giuseppe Cicala, Luca Pulina, and Armando Tacchella. An experimental evaluation of tools for ontology-based data access. In *Proceedings of the 20th RCRA International Workshop on Experimental Evaluation of Algorithms for solving problems with combinatorial explosion*, 2013. Avalaible on-line from `https://docs.google.com/file/d/0B8dEUbPKR1laYjFnMUJLcks0ZnM/edit?usp=sharing`.
5. Jean-Remi Bourguet, Giuseppe Cicala, Luca Pulina, and Armando Tacchella. Obda and intermodal logistics: Active projects and applications. In *Web Reasoning and Rule Systems (RR) 2013*, volume 7994 of *LNCS*, pages 210–215. Springer Verlag, 2013.
6. Francesco Calimeri, Giovambattista Ianni, Francesco Ricca, Mario Alviano, Annamaria Bria, Gelsomina Catalano, Susanna Cozza, Wolfgang Faber, Onofrio Febbraro, Nicola Leone, et al. The third answer set programming competition: Preliminary report of the system competition track. In *Logic Programming and Nonmonotonic Reasoning*, pages 388–403. Springer, 2011.
7. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, and R. Rosati. Ontologies and Databases: The *DL-Lite* approach. *Reasoning Web. Semantic Technologies for Information Systems*, pages 255–356, 2009.
8. Matteo Casu, Giuseppe Cicala, and Armando Tacchella. Ontology-based data access: An application to intermodal logistics. *Information Systems Frontiers*, 2012.
9. Thomas Eiter, Magdalena Ortiz, M Simkus, Trung-Kien Tran, and Guohui Xiao. Towards practical query answering for horn-shiq. *Description Logics*, 846, 2012.
10. Birte Glimm et al. OWL-BGP – A framework for parsing SPARQL basic graph patterns (BGPs) into an OWL object representation. `http://code.google.com/p/owl-bgp`.
11. Achille Fokoue, Thorsten Liebig, Eric Goodman, Jesse Weaver, Jacopo Urbani, and David Mizell. Joint workshop on scalable and high-performance semantic web systems (ssws+ hpcsw 2012). 2012.
12. Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing shiq- description logic to disjunctive datalog programs. *Proc. KR*, 4:152–162, 2004.
13. Ilianna Kollia, Birte Glimm, and Ian Horrocks. Sparql query answering over owl ontologies. In *The Semantic Web: Research and Applications*, pages 382–396. Springer, 2011.
14. Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for owl-dl with rules. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60, 2005.
15. Boris Motik, Rob Shearer, and Ian Horrocks. Optimized reasoning in description logics using hypertableaux. *Automated Deduction–CADE-21*, pages 67–83, 2007.
16. Héctor Pérez-Urbina, Ian Horrocks, and Boris Motik. Efficient query answering for owl 2. In *The Semantic Web-ISWC 2009*, pages 489–504. Springer, 2009.

17. Claudia Peschiera, Luca Pulina, Armando Tacchella, Uwe Bubeck, Oliver Kullmann, and Inês Lynce. The seventh qbf solvers evaluation (qbfeval10). In *Theory and Applications of Satisfiability Testing–SAT 2010*, pages 237–250. Springer, 2010.
18. E. Prud'Hommeaux and A. Seaborne. SPARQL Query Language for RDF. *W3C working draft*, 4(January), 2008.
19. M. Rodriguez-Muro and D. Calvanese. High Performance Query Answering over DL-Lite Ontologies. In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 308–318, 2012.
20. M. Rodrıguez-Muro and D. Calvanese. Quest, an OWL 2 QL Reasoner for Ontology-based Data Access. *OWLED 2012*, 2012.
21. A. Seaborne. ARQ – A SPARQL Processor for Jena, 2010. `http://jena.sourceforge.net/ARQ/` – [accessed 1/5/2010].
22. Rob Shearer, Boris Motik, and Ian Horrocks. Hermit: A highly-efficient owl reasoner. In *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2008)*, pages 26–27, 2008.
23. E. Sirin, B. Parsia, B. Cuenca-Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007. Available on-line from `http://pellet.owldl.com/`.
24. Geoff Sutcliffe. The cade-23 automated theorem proving system competition–casc-23. *AI Communications*, 25(1):49–63, 2012.
25. Edward Thomas, Jeff Z. Pan, and Yuan Ren. TrOWL: Tractable OWL 2 Reasoning Infrastructure. In *the Proc. of the Extended Semantic Web Conference (ESWC2010)*, 2010.