Analysing the Trade-Off Between Computational Performance and Representation Richness in Ontology-Based Systems

Salvatore F. Pileggi[1(✉)], Fabian C. Peña[2], Maria Del Pilar Villamil[2], and Ghassan Beydoun[1]

[1] School of Information, Systems and Modelling, University of Technology Sydney, Ultimo, Australia
{SalvatoreFlavio.Pileggi,Ghassan.Beydoun}@uts.edu.au
[2] Systems and Computing Engineering Department, School of Engineering, Universidad de los Andes, Bogota, Colombia
{fc.pena,mavillam}@uniandes.edu.co

Abstract. As the result of the intense research activity of the past decade, Semantic Web technology has achieved a notable popularity and maturity. This technology is leading the evolution of the Web via interoperability by providing structured metadata. Because of the adoption of rich data models on a large scale to support the representation of complex relationships among concepts and automatic reasoning, the computational performance of ontology-based systems can significantly vary. In the evaluation of such a performance, a number of critical factors should be considered. Within this paper, we provide an empirical framework that yields an extensive analysis of the computational performance of ontology-based systems. The analysis can be seen as a decision tool in managing the constraints of representational requirements versus reasoning performance. Our approach adopts synthetic ontologies characterised by an increasing level of complexity up to OWL 2 DL. The benefits and the limitations of this approach are discussed in the paper.

Keywords: Semantic web · Semantic technology · Ontology · Computational performance

1 Introduction

The Semantic Web [3] has achieved a notable popularity as a mature technological environment. In big part, this is due the intense research activity of the last 15 years, and the efforts of W3C[1] to promote a standardisation process for the different languages and their underlining models. Semantic technologies are leading the evolution of the Web via interoperability by providing structured

[1] World Wide Web Consortium (W3C) - https://www.w3.org.

J. M. F. Rodrigues et al. (Eds.): ICCS 2019, LNCS 11540, pp. 237–250, 2019.
https://doi.org/10.1007/978-3-030-22750-0_18

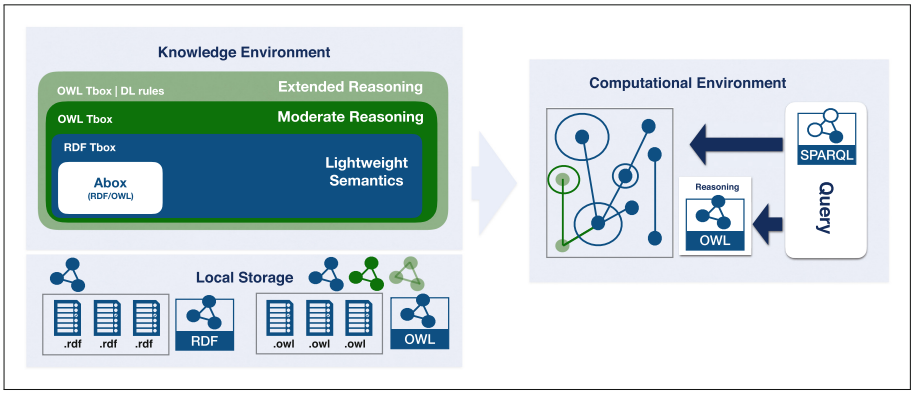

Fig. 1. Computational model.

metadata. Because of the adoption of rich data models on a large scale to support the representation of complex relationships among and standard reasoning, the computational performance of ontology-based systems may be hard to evaluate, as it may change significantly depending on the application context. A number of critical and key factors should be considered. Firstly, the Semantic Web technology provides a technological ecosystem composed of several languages. These languages are characterised by an increasing complexity to support different data modelling spaces. Secondly, applications may propose very different behaviours and may consequently adopt the technology in different ways.

This paper provides a performance evaluation framework for ontology-based systems supported by empirical measurements. The proposed framework takes into account the perennially conspicuous trade-off between computational performance and representational capabilities. Our analysis is limited to decidable technology, including *lightweight semantics* based on RDF [5] reasoning, *moderate reasoning* equivalent to OWL-Lite [2] reasoning and *extended reasoning* corresponding to OWL-DL [2]. OWL ontologies are implemented in OWL 2.

2 Related Work

The analysis of the trade-off between computational performance and representation richness is a classic topic extensively reported in literature. A number of OWL benchmarks are compared in [16], where also the specification of a set of requirements for an ideal OWL benchmark is provided.

Similar approaches are followed also to compare different reasoners in other contributions (e.g. [1,4,6,7]). An interesting comparison between two of the most relevant computation techniques (tableau and hyper-tableau calculus) is proposed in [13]. One of the most popular OWL benchmarks is LUBM [10], which provides advanced analysis features to evaluate systems characterized by different reasoning capabilities and storage mechanisms. It addresses generic OWL data-spaces and approaches performance analysis by providing global metrics

suitable to compare different systems. More recently, a competition based on a testing framework agreed within the community has been arranged [14].

Our work differs from those mentioned as we provide an environment suitable to multi-dimensional analysis in which the performance of a given system may be evaluated as the function of the ontology complexity and its scale (population). These two dimensions are addressed by generating synthetic ontologies (Sect. 3.4) which enable fine-grained analysis. Our approach assures a generic and an application-independent performance analysis that relies on the specification of complexity ranges (Sect. 3) and on computational experimentation. Furthermore, we aims at providing domain and architecture agnostic results by introducing a number of simplifications (see Sect. 3.1). For instance, we don't take into account architectural (e.g. storage system) and network factors, as well as we adopt a query-independent approach. Those simplifications allow a more focused, direct and understandable analysis framework. Last but not the least, our framework is extensible, meaning that further dimensions of analysis may easily be addressed.

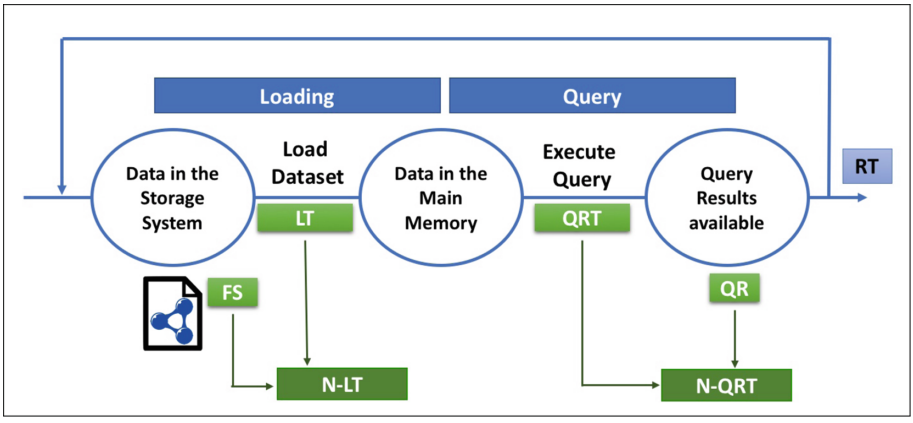

Fig. 2. Experiment phases and metrics associated.

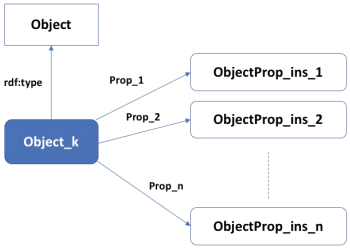

Fig. 3. Synthetic object.

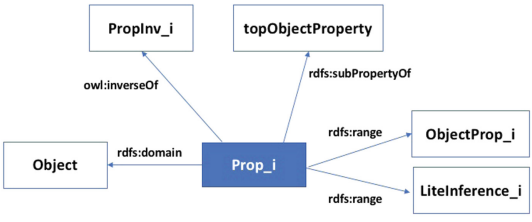

Fig. 4. Synthetic property.

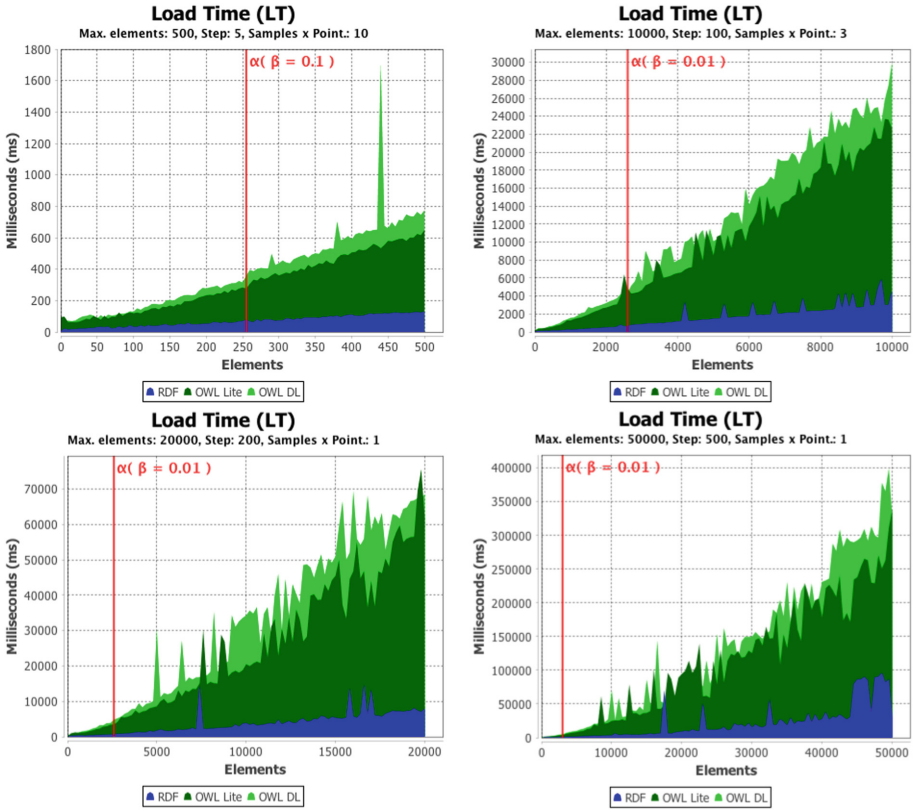

Fig. 5. *Load Time* measured for some of the experiments performed.

3 Evaluation Framework

Within our framework, we define three different levels of ontology complexity (Fig. 1) as follows:

- **Lightweight semantics.** We associate lightweight semantics with a minimal set of knowledge representation requirements and, therefore, with the best computational performance. We assume RDF [5] structures and reasoning.
- **Moderate reasoning.** The most immediate extension for lightweight semantics as previously defined is to provide more extended reasoning capabilities, to uncover non-explicit relations among basic concepts. We associate this level of complexity with OWL and, more concretely, with OWL-Lite [2] complexity. Such a step forward introduces additional constructs and abstractions (e.g. data and object property), structural relations (e.g. class/sub-class and property/sub-property), constraints (e.g. class disjointedness, functional property), relations among properties (e.g. inverse properties) and basic inference on properties (domain and range).

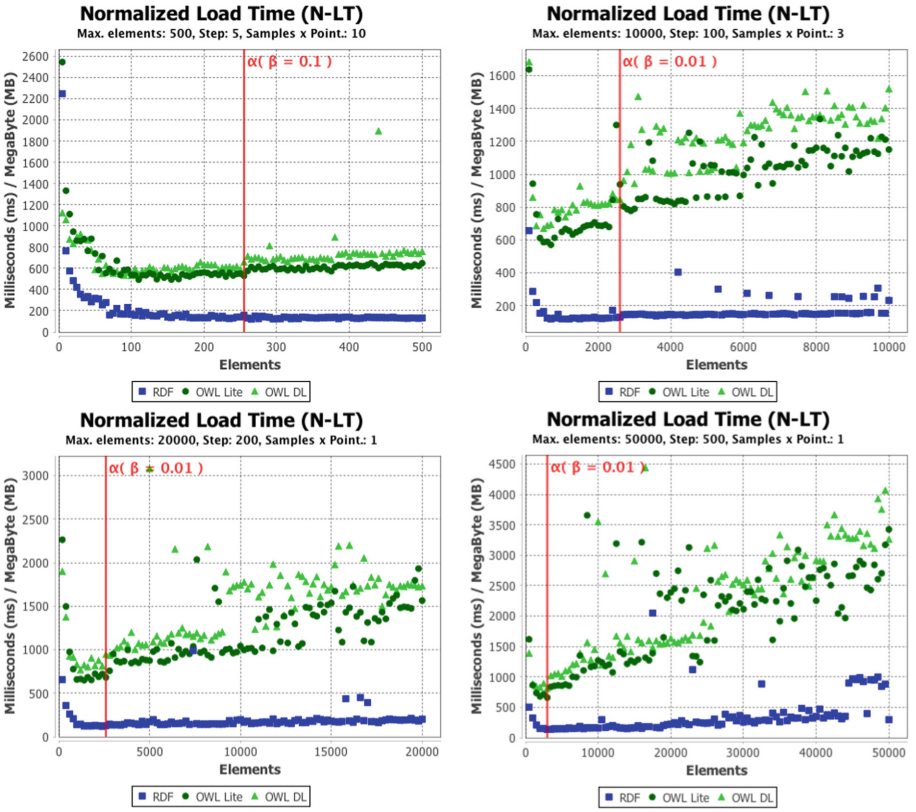

Fig. 6. *Normalized Load Time* measured for some of the experiments performed.

– **Extended reasoning capabilities.** The highest level of complexity that we consider within this work corresponds to OWL-DL [2], which assures the maximum expressiveness maintaining computational completeness and decidability. This level of complexity extends the previous one by providing the capability to define inference rules according to a Description Logic. This extension results in more advanced reasoning capabilities.

3.1 Assumptions and Simplifications

A comprehensive study on the computational performance of ontology-based systems should consider several factors. For simplicity sake, we consider a simplified, still in our opinion exhaustive, environment, adopting the following assumptions:

– **Local storage (file system).** The Semantic Web is a distributed environment by definition. In a Web context, data can be potentially retrieved from multiple, eventually remote, sources. Moreover, target data could be stored in files, normally accessible by URLs, as well as in common databases or even

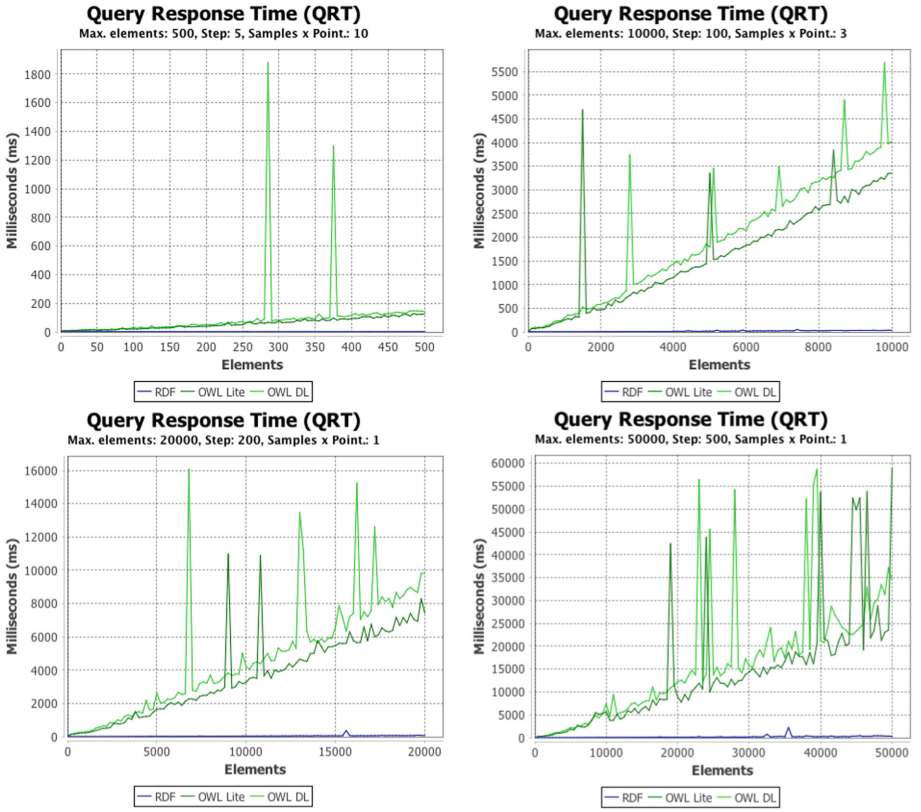

Fig. 7. *Query Response Time* for some of the experiments performed.

specialised data-stores (triple-stores). In this work, we consider uniquely local storage in the file system. This allows an analysis independent from the performance of the storage system.

– **Query-independent evaluation.** A study that takes into account the complexity of the query would be very interesting. However, it would add a significant complexity. To assure a query-independent evaluation, we consider the generic SPARQL query below:

```
PREFIX onto: ourPrefix
SELECT ?x ?y
WHERE {    ?x a ?y .
      FILTER regex( str( ?x ), ourPrefix ) .
      FILTER regex( str( ?y ), ourPrefix )
}
```

This query can be applied to both RDF and OWL environments and returns all the elements that are member of some class. As will be later explained, this allows to clearly identify the contribution of inference to the query outcome.

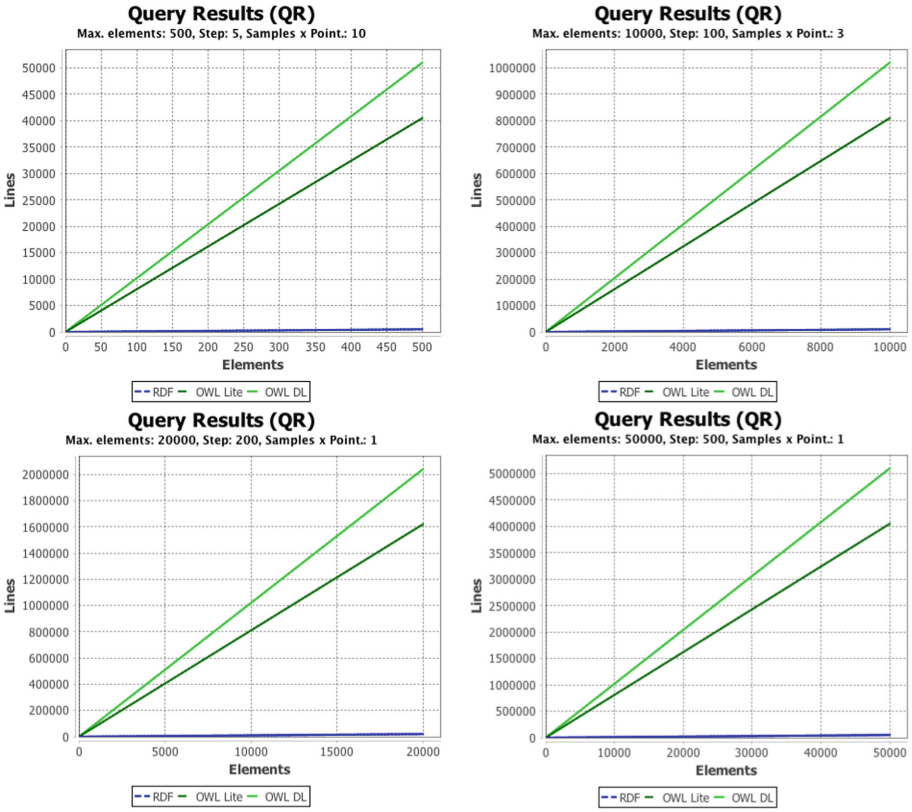

Fig. 8. *Query Response* for some of the experiments performed.

- **Ontology size increases by increasing its population only (Abox).** We consider two main dimension of analysis: the ontology complexity and the ontology size. The former, associated with the Tbox including inference rules, is defined by a number of templates as a kind of static configuration. The latter is associated with the Abox and is dynamically addressed.
- **Agnostic approach to software components.** We consider the semantic engine (reasoner) as a black-box. That is, we have designed our framework on the basis of macro-operations common to all common APIs in semantic technology. For our experiments, without loss of generality, we only use HermiT [15]. Naturally, we can perform the same experiments using any other semantic engines supporting RDF and OWL2 DL.
- **Synthetic Ontology.** In order to provide a fine-grained analysis, we opt for an environment which produces synthetic ontologies according to common approaches [12]. We believe the experimentation on real ontologies is not a relevant factor within the scope of this work. However, as briefly discussed later on, it may introduce some uncertainty in the analysis.

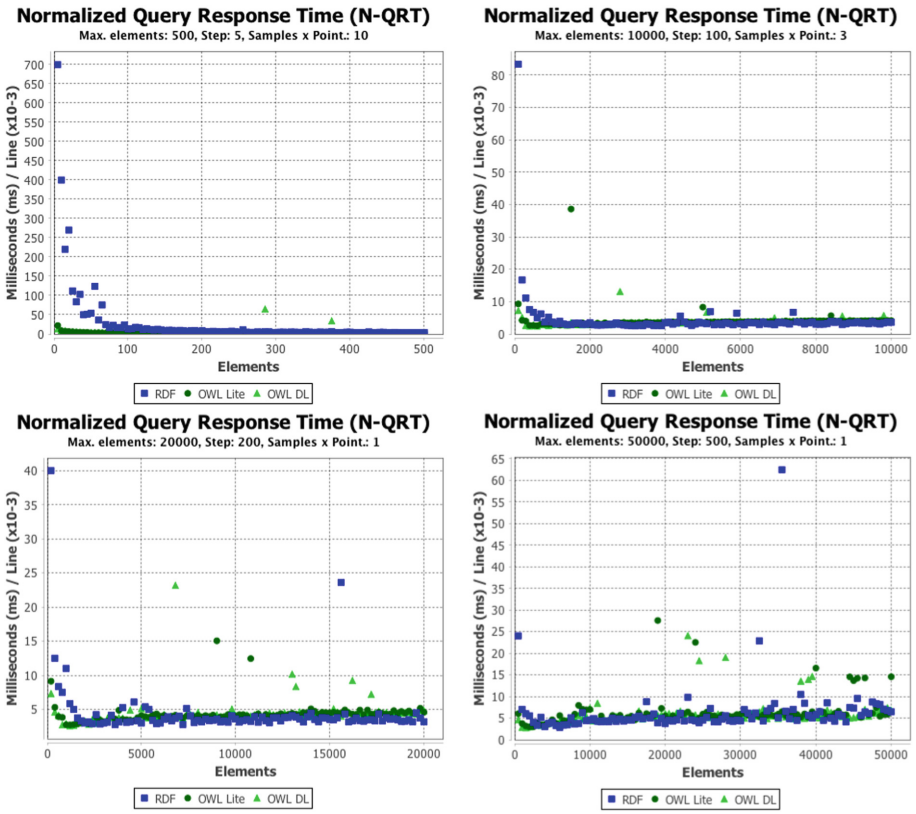

Fig. 9. *Normalized Query Response Time* for some of the experiments performed.

3.2 From Lightweight Semantics to Complex Reasoning

We approach the computational performance evaluation of ontology-based systems according to a classic perspective, which takes into account two major macro-operations: *loading the information* into the semantic engine and *executing a query* on the information available (Fig. 2).

Control Mechanism for the Input Dataset. In order to control eventual gaps between logical and physical representations (e.g. RDF and OWL implementations), we introduce a simple control mechanism for input datasets. The *convergence point* (α) is defined as the function of a parameter, the *convergence threshold* (β); for instance, $\alpha_{(\beta=1\%)} = 100000$ means that, for a scale higher than 100000 atomic elements, the difference in size for the considered representations is within the 1% of the smallest size. β is normally expressed as a percentage of the shorter representation size and is normally supposed to be a small value. Such a metric may be very relevant for experiments involving small datasets. Indeed, it expresses the end of the transitory and the beginning of the stationary condition for a given experiment: while those experiments at a lower

scale than α (transitory) are affected by the difference in size of the considered representations, the experiments at an higher scale (stationary condition) are assumed not affected by such overheads.

3.3 Metrics

The experiment performed are modelled as an iterative process (Fig. 2). After each iteration, the size of the input dataset is increased. Each iteration is composed of two different phases: in the first, a file of size FS is loaded from the storage systems into the main memory (loading phase); in the second, a query on the available dataset is executed (query phase).

To assess the computational performance in each phase, we consider the three following metrics:

- **Load Time (LT)** is the time needed to load the dataset from the storage systems into the main memory.
- **Query Response Time (QRT)** is the execution time for a query. QRT assumes the target dataset already loaded in the main memory.
- **Query Results (QR)** is the number of rows of the result set returned by a given query.

In order to have a concise assessment of the performance, we define two further normalized metrics based on the three defined above:

- **Normalized Load Time (N-LT)** is defined as LT/FS. Within our evaluation framework, $N - LT$ concisely expresses loading performance because the load time is considered as the function of the dataset size.
- **Normalized Query Response Time (N-QRT)** is defined as QRT/QR. $N - QRT$ reflects the query performance: it provides an understanding of the query response time as the function of the query result set size.

For completeness, we also define a global metric, *Response Time (RT)*, which is the sum of the Load Time and of the Query Response Time ($LT + QRT$). However, In order to assure a consistent analysis, RT should be considered both with and in the context of the normalized metrics previously discussed.

3.4 Synthetic Patterns

As earlier mentioned, the scale of the ontologies adopted in the experiments is enlarged by increasing its population (Abox). The synthetic object adopted to populate the Abox is represented in Fig. 3: it is declared to be an instance of the class *Object* and it is related to n static objects through an equivalent number of properties.

We provide moderate reasoning by defining a number of Object Properties according to the model depicted in Fig. 4. Each object property is a sub-property

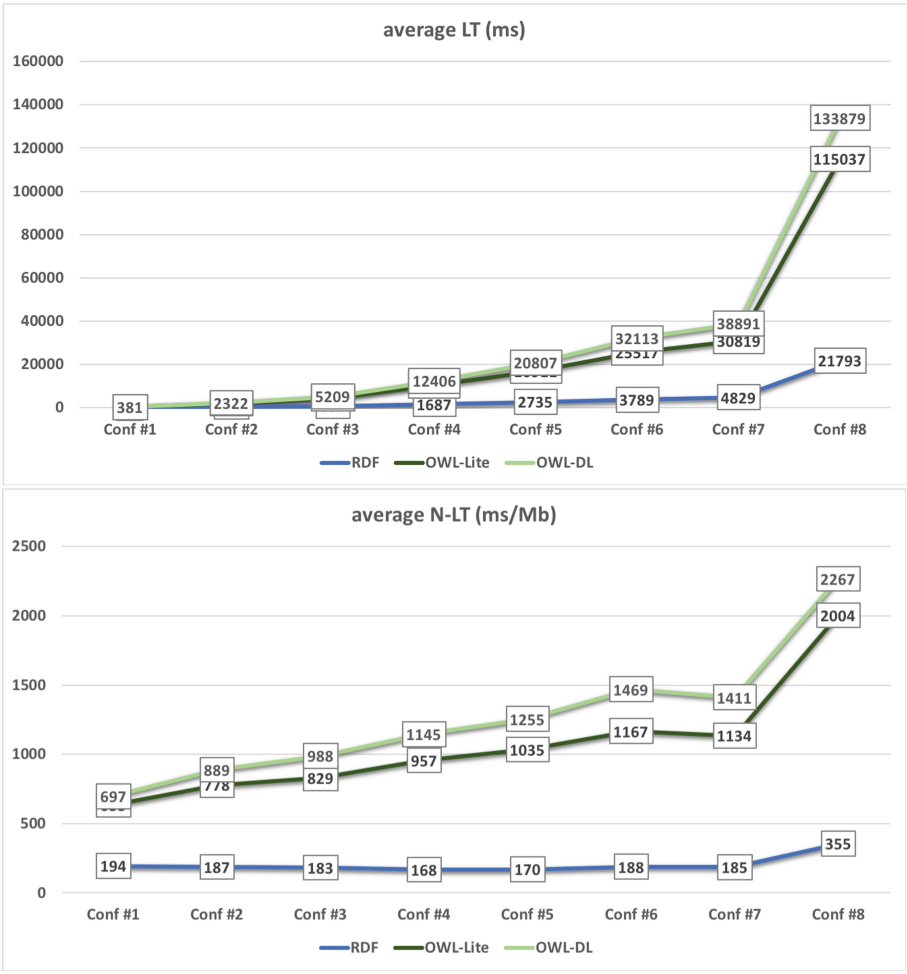

Fig. 10. Loading performance (average values).

of *topObjectProperty*. Its domain and range are defined according to RDF-S specifications. For each object property, an additional property is defined and declared as an inverse of the considered property, according to OWL specifications.

Finally, extended reasoning capabilities are provided by a set of DL statements that define equivalent classes according to the pattern reported in Eq. 1. The DL statement is composed of two different sub-statements adopting the operator *some*; those two sub-statements are related by the operator OR or AND.

$$(Prop_i \ some \ ObjectProp_i) \quad AND|OR \quad (Prop_{i+1} \ some \ ObjectProp_{i+1}) \quad (1)$$

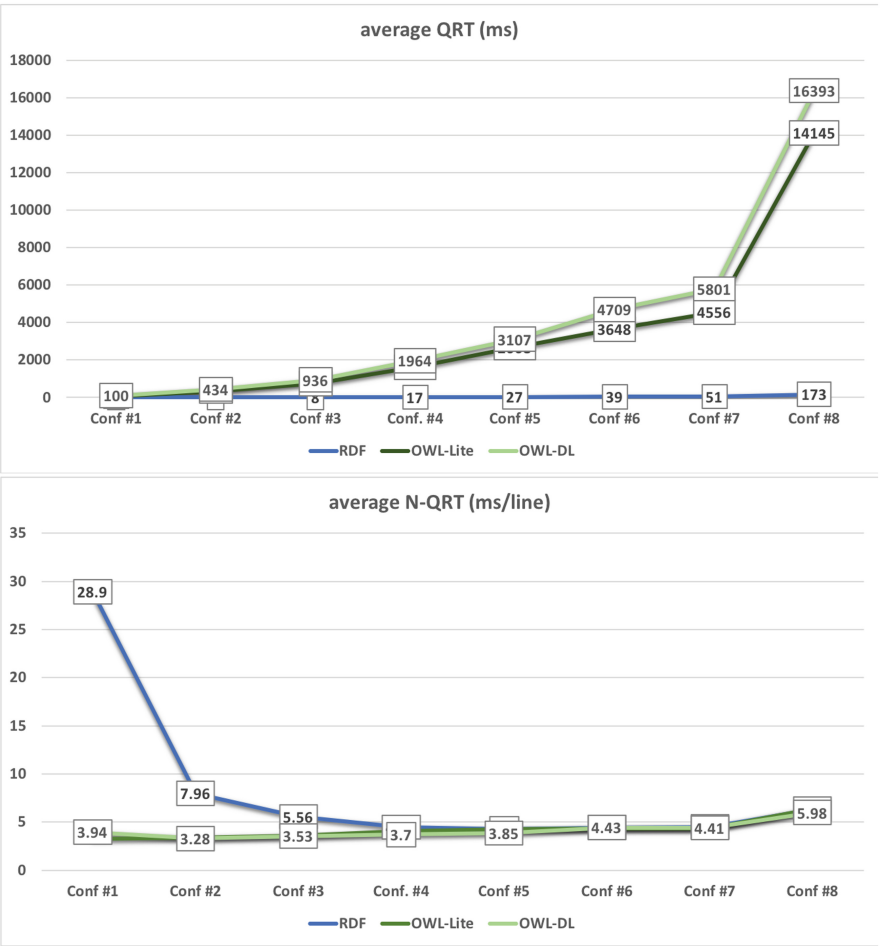

Fig. 11. Query performance (average values).

4 Experimental Performance Evaluation

We adopt commonly accepted metrics from Protege [8] to measure the complexity of the ontologies considered. The different configurations corresponding to non-populated ontologies (Tbox) are reported in Table 1. We select three different perspectives by considering *axioms*, *classes* and *object properties*. We distinguish between logical and declaration axioms. We also report the number of sub-classes and equivalent classes defined by DL statements. For properties, we consider the number of inverse properties and the statements associated with the object property domain and range.

Our experimentation consists of a number of empirical measurements for the different ontologies by increasing the Abox size as in the configurations reported in Table 2.

Table 1. Ontology configuration.

Ontologies configuration (Tbox)									
Ont. complexity	Axioms			Classes			Object properties		
	Tot	Log.	Decl.	Tot	SubC	Equiv.	Inv.	Dom.	Ran.
Lightweight (≈ RDF)	21	1	0	**1**	0	**0**	0	0	0
Moderate (≈ OWL-Lite)	301	196	105	**43**	40	**0**	20	20	40
Extended (≈ OWL-DL)	362	236	126	**64**	60	**20**	20	20	40

Table 2. Experiments configuration.

Conf.	Experiment configuration				
	Elements	File size (MB)	Step	Samples x point	β
1	0-500	up to 1	5	10	0.1
2	0-2500	up to 5	25	5	0.05
3	0-5000	up to 10	50	5	0.05
4	0-10000	up to 20	100	3	0.01
5	0-15000	up to 30	150	2	0.01
6	0-20000	up to 40	200	1	0.01
7	0-25000	up to 50	250	1	0.01
8	0-50000	up to 100	500	1	0.01

For example, the experiment #1 considers files with logical elements in the range 0–500, namely files of a size up to 1 MB; the experiment starts with a file of the minimum size (5 elements in this case); the file is increased of a step of 5; each measure reported is the average over 10 independent samples; β is 0.1. We adopt a software engine based on Hermit [9,15] as a reasoner and OWL-BGP[2] [11] as SPARQL wrapper. The framework is developed in Java. All the experiments reported in the paper have been executed on a common laptop (1.8 GHz Intel Core i5, 8 GB 1600 MHz DDR3, macOS Sierra). For each experiment as defined in Table 2, the measurements are executed sequentially without re-booting.

The *Load Time (LT)* measured for some of the experiments performed is reported in Fig. 5. Likewise, the *Normalized Load Time (N-LT)* is shown in Fig. 6. Similarly we report the metrics for query performance evaluation: Figs. 7, 8 and 9 show respectively the *Query Response Time (QRT)*, the *Query Response (QR)* and the *Normalized Query Response Time (N-QRT)*.

As previously explained, normalised metrics take into account both the size of the dataset imported and the size of the result set. Therefore, the metrics provide a concise measure of loading and query performance, allowing comparison among the different languages. We design our experiments to minimize the impact of size variations for the input datasets (Sect. 3.2). Indeed, *LT* and *N-LT* present a similar pattern. Average values for all experiments are reported in Fig. 10. Looking exclusively at loading performance, RDF clearly outperforms OWL. That is because the reasoner adopted implements hyper-tableau calculus. A minor difference between OWL-Lite and OWL-DL is also detected.

[2] OWL-BGP - https://github.com/iliannakollia/owl-bgp.

Looking at query performance, average values for *QRT* and *N-QRT* are reported in Fig. 11. *QRT* highlights variations in performance across the different models considered throughout the range of experiments. The pattern detected for *N-QRT* is quite interesting as it clearly shows the computation performance of hyper-tableau calculus in normalized conditions. Indeed, according to this normalized metric that takes into account the contribution of inference in terms of query output, at a significant scale there is no difference of performance among the three levels of complexity.

5 Main Limitations and Uncertainty

The simplifications introduced in the framework (Sect. 3.1) have allowed a systematic, fine-grained and relatively simple analysis in stable and normalized conditions. On the other hand, such an approach may introduce a number of possible uncertainty factors to the key question on performance evaluation of real ontology-based systems.

The very first factor of uncertainty is the use of synthetic ontologies. They are designed around a number of typical design patterns. Real ontologies may include those patterns or a part of them, as well as they may propose completely different ones. Additionally, the distributed approach to modern systems may introduce key trade-offs beyond network factors, such as between cloud and edge computing along a wide range of hybrid solutions (e.g. fog computing). Similar considerations affect key architectural components, such as the storage system. Indeed, the design of the whole architecture needs to be considered depending on the data consistency model (e.g. weak and strong consistency).

6 Conclusions and Future Work

In this paper, we define a performance evaluation framework for ontology-based systems in which ontology complexity and ontology size are considered as the main dimensions for performance analysis. In essence, our framework can be used to ensure the right mix of responses to the constraints imposed by the trade-off between reasoning computation and knowledge expressiveness requirements (ranging from lightweight semantics, moderate reasoning and extended reasoning capabilities respectively).

We introduced a number of simplifying assumptions (discussed in Sect. 3.1) to enable a relatively simple environment for the analysis of computational performance. We believe however that our metrics reflect relatively realistic working conditions. They combine simplicity and coverage leading to a direct and viable analysis. We have performed a number of experiments at a relatively low scale, involving files up to 100 MB. Without loss of generality, we fixed the semantic engine across all experiments and opted for uniform use of a single reasoned. Comparing various reasoners has been undertaken elsewhere. We thus focused on the definition of a framework which allows generic and effective performance

analysis by considering increasing capabilities in terms of representation and reasoning. With the increasing prominence of IoT and AI based applications, the trade off between complexity in representation and performance is a more pressing concern for many innovations. This is particularly true for distributed settings. Hence, in future work, we will consider distributed environments, namely data sets imported by different remote sites and performance analysis as the function of the ontology Tbox.

References

1. Abburu, S.: A survey on ontology reasoners and comparison. Int. J. Comput. Appl. **57**(17), 33–39 (2012)
2. Bechhofer, S.: OWL: web ontology language. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, pp. 2008–2009. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-39940-9
3. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The semantic web. Sci. Am. **284**(5), 28–37 (2001)
4. Bock, J., Haase, P., Ji, Q., Volz, R.: Benchmarking OWL reasoners. In: ARea2008-Workshop on Advancing Reasoning on the Web: Scalability and Commonsense. Tenerife (2008)
5. Decker, S., et al.: The semantic web: the roles of XML and RDF. IEEE Internet Comput. **4**(5), 63–73 (2000)
6. Dentler, K., Cornet, R., Ten Teije, A., De Keizer, N.: Comparison of reasoners for large ontologies in the OWL 2 EL profile. Semant. Web **2**(2), 71–87 (2011)
7. Gardiner, T., Tsarkov, D., Horrocks, I.: Framework for an automated comparison of description logic reasoners. In: Cruz, I., et al. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 654–667. Springer, Heidelberg (2006). https://doi.org/10.1007/11926078_47
8. Gennari, J.H., et al.: The evolution of Protégé: an environment for knowledge-based systems development. Int. J. Hum. Comput. Stud. **58**(1), 89–123 (2003)
9. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: an OWL 2 reasoner. J. Autom. Reason. **53**(3), 245–269 (2014)
10. Guo, Y., Pan, Z., Heflin, J.: LUBM: a benchmark for OWL knowledge base systems. In: Web Semantics: Science, Services and Agents on the World Wide Web, vol. 3(2), pp. 158–182 (2005)
11. Kollia, I., Glimm, B.: Optimizing SPARQL query answering over OWL ontologies. J. Artif. Intell. Res. **48**, 253–303 (2013)
12. Link, V., Lohmann, S., Haag, F.: OntoBench: generating custom OWL 2 benchmark ontologies. In: Groth, P., et al. (eds.) ISWC 2016, Part II. LNCS, vol. 9982, pp. 122–130. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46547-0_13
13. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. Artif. Intell. Res. **36**(1), 165–228 (2009)
14. Parsia, B., Matentzoglu, N., Gonçalves, R.S., Glimm, B., Steigmiller, A.: The OWL reasoner evaluation (ORE) 2015 competition report. J. Autom. Reason. **59**(4), 455–482 (2017)
15. Shearer, R., Motik, B., Horrocks, I.: HermiT: a highly-efficient OWL reasoner. In: OWLED, vol. 432, p. 91 (2008)
16. Weithöner, T., Liebig, T., Luther, M., Böhm, S.: What's wrong with OWL benchmarks. In: Proceedings of the Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006), pp. 101–114. Citeseer (2006)