Context-Dependent OWL Reasoning in Sindice - Experiences and Lessons Learnt *

Renaud Delbru[1], Giovanni Tummarello[1], and Axel Polleres[1,2]

[1] Digital Enterprise Research Institute
National University of Ireland, Galway
{renaud.delbru,giovanni.tummarello,axel.polleres}@deri.org
[2] Siemens AG Österreich
Siemensstrasse 90, 1210, Vienna, Austria

Abstract. The Sindice Semantic Web index provides search capabilities over 260 million documents. Reasoning over web data enables to make explicit what would otherwise be implicit knowledge: it adds value to the information and enables Sindice to ultimately be more competitive in terms of precision and recall. However, due to the scale and heterogeneity of web data, a reasoning engine for the Sindice system must (1) scale out through parallelisation over a cluster of machines; and (2) cope with unexpected data usage. In this paper, we report our experiences and lessons learned in building a large scale reasoning engine for Sindice. The reasoning approach has been deployed, used and improved since 2008 within Sindice and has enabled Sindice to reason over billions of triples.

1 Introduction

Reasoning over semantic entity description enables to make explicit what would otherwise be implicit knowledge: it adds value to the information and enables a web data search engine such as Sindice to ultimately be more competitive in terms of precision and recall [16]. The drawback is that inference can be computationally expensive, and therefore drastically slow down the process of indexing large amounts of information. Therefore, large scale reasoning through parallelisation is one requirement of Sindice.

A common strategy for reasoning with web data is to put several entity descriptions together and to compute the deductive closure across all the entity descriptions. However, we can not expect web data to always adhere to strict rules. Web data is highly heterogeneous and unexpected usage of data and data schema is common. For example, data can be erroneous or crafted for malicious purposes. As a consequence, there is a risk for a reasoner to infer undesirable logical assertions which is harmful for the Information Retrieval system. These assertions increase the noise in the data collection and decrease the precision of the system, losing the benefits that reasoning should provide. In addition, such inferences add an unnecessary computational overhead which

* A preliminary version [6] of this article was presented at the 4th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2008). We have extended it with a comparison with other large scale reasoning approaches, a performance evaluation, and reports on using and optimising the presented reasoning approach in a production system – the Sindice Semantic Web index – since 2008.

augments the demand of computational resources and limits the performance of the system. Therefore, a second requirement of inference engines for web data is the ability to cope with disparate data quality.

For these reasons, a fundamental requirement has been to confine T-Box assertions and reasoning tasks into "contexts" in order to track the provenance of inference results. By tracking the provenance of each single T-Box assertion, we are able to prevent one ontology to alter the semantics of other ontologies on a global scale. In addition, such a context-dependent approach provides an efficient distributed computing model which scales linearly with the amount of data.

Section 2 provide an overview of the current approaches for large scale reasoning over web data. In Section 3, we introduce the core concepts our approach. Section 4 describes a context-dependent TBox, called an ontology base. The conceptual model of the ontology base is detailed and our query and update strategies are discussed. Section 5 discusses the distributed implementation and Section 6 an empirical evaluation before concluding in Section 8.

2 Reasoning over Web Data

In this section, we first provide an overview of the current approaches for large scale reasoning over web data. Next, we review existing contextual reasoning approaches and other works related to our context-dependent reasoning technique. Finally, we introduce the concept of *Context* on the Semantic Web and its formal framework as it is defined by Guha [10]. This framework is used in the development of our context-dependent reasoning mechanism.

2.1 Distributed Reasoning

Recently, we have seen major interests towards large scale reasoning based on parallel distribution of inference. [25, 24] present parallel RDFS reasoning. In [23], they extend their approach to cover the OWL Horst fragment [14]. SAOR [12, 13] aims to cover a larger subset of OWL2RL.

In these approaches, the separation of terminological data from assertional data, which are commonly known as the *T-Box* and *A-Box* respectively, is a key point for parallel inference. The separation of T-Box from A-Box allows to distribute reasoning among a cluster of machines more easily. Since the T-Box is relatively small, the T-Box is replicated to each cluster node. The A-Box is partitioned and distributed among the nodes. Each node calculates the closure of its A-Box partition.

With respect to computational optimisation, [24, 12, 13] have identified properties about the ordering of rules in order to avoid multiple passes over the data when possible. In addition, [23] proposes a fixpoint iteration to compute inference of rules that requires multiple A-Box joins. SAOR avoids rules which require A-Box joins and consequently is less complete than [23]. [25] does not need A-Box joins since it is restricted to RDFS where there are no dependencies between partitions. With respect to IO load, [23, 12, 13] which deal with richer logics than RDFS necessitate generally more than one pass

over the data. This means that the data is read multiple times from disk. For very large data collection, this is time and resource consuming.

In our context-dependent reasoning technique, we also separate T-Box from A-Box, but compared to the previous approaches, we partition and distribute the A-Box on a per context fashion. The statements in each A-Box partition belong to a same context, e.g., a document and its related documents through implicit or explicit imports. Each A-Box partition is small enough to be fully loaded in memory. We can then perform conventional reasoning in memory and avoid to write and read intermediate results on disk.

Although [25, 24, 23] have demonstrated to scale in the order of hundreds of millions or hundreds of billions triples, their experiments are focussed on scalability issues, disregarding the problem of data quality and its consequences. In [12], Hogan et al show evidences of a large number of unexpected inferences with real-world data. SAOR proposes a technique called *Authoritative Reasoning* to avoid undesirable inference results. Their system performs an analysis of the authority of web data sources, and imposes rule restrictions if non-authoritative T-Box statements are detected. Our context-dependent reasoning technique adopts a completely different approach against the problem of noisy data and undesirable inferences. Instead of imposing rule restrictions, we consider a context as a closed world where inferences are "quarantined" within the context.

2.2 Contextual Reasoning

The notion of context has been extensively studied since the early 1990s, starting with a first formalisation by Guha [9] followed by McCarthy [17] and by Giunchiglia [8]. For more information about the domain, [20] presents a survey and comparison of the main formalizations of context. There are also recent works that adapt context formalisations to the Semantic Web such as [3, 10].

The previous references provide a strong theoretical background and framework for developing a context-dependent reasoning mechanism for the Semantic Web. To the best of our knowledge, only [22] proposes a concrete application of management of contextualized RDF knowledge bases. But no one has described a methodology for efficient large scale reasoning that deals with contextuality of web documents.

Assumption-based Truth Maintenance Systems [15] are somehow comparable to our approach since such a system keeps track of dependency between facts in multiple views (contexts) in order to maintain consistency in a knowledge base. The difference with our approach lies in that we do not maintain consistency, instead we just maintain the provenance of each inferred T-Box statement.

A side remark in [7] suggests that the authors follow a very similar strategy to ours in determining the ontological closure of a document (see Section 3.1), but no details on efficient contextual confinement and reuse of inferences, which are the main contributions of this work, are discussed.

2.3 Contexts on the Semantic Web

The URI of an entity description provides a natural way to define a context for the data it contains. Naming an RDF graph has multiple benefits [4]. It helps tracking the provenance of each statement. In addition, since named graphs are treated as first class objects, this enables the description and manipulation of a set of statements just as for any other resource. In this work, the notion of context refers to the entity URI at which the entity description is retrievable.

Guha [10] proposed a context mechanism for the Semantic Web which provides a formal basis to specify the aggregation of contexts. Within his framework, a context denotes the scope of validity of a statement. This scope is defined by the symbol ist ("is true in context"), introduced by Guha in [9]. The notation $ist(c, \varphi)$ states that a proposition φ is true in the context c.

3 Context-Dependent Reasoning of RDF Models

It would be practically impossible, and certainly incorrect, to apply reasoning over a single model composed of all the data found on the Web. Letting alone the computational complexity, the problem here is clearly the integration of information from different sources: data is published in a particular context, and a naive integration of information coming from different contexts can result in undesirable inferences. We therefore ensure that the context is maximally preserved when reasoning with web data.

To reason on contexts, we assume that the ontologies that these contexts refer to are either included explicitly with `owl:imports` declarations or implicitly by using property and class URIs that link directly to the data describing the ontology itself. This later case should be the standard if the W3C best practices [18] for publishing ontologies and the Linked Data principles [2] are followed by data publishers. As ontologies might refer to other ontologies, the import process then needs to be recursively iterated as explained in Section 3.1.

A naive approach would be to execute such a recursive fetching for each entity description and to create an aggregate context (see Section 2.3 for a definition) composed by the original description plus the imported ontologies. At this point the deductive closure of the aggregate context can be computed as explained in Section 3.2. Such a naive procedure is however obviously inefficient since a lot of processing time will be used to recalculate the T-Box deductions which could be instead reused for possibly large numbers of other entity descriptions. Therefore, we propose a mechanism to store and reuse such deductions in Section 4.

3.1 Import Closure of RDF Models

On the Semantic Web, ontologies are published in order to be easily reused by third parties. OWL provides the `owl:imports` primitive to indicate the inclusion of a target ontology inside an RDF model. Conceptually, importing an ontology brings the content of that ontology into the RDF model.

The `owl:imports` primitive is transitive. That is, an import declaration states that, when reasoning with an ontology O, one should consider not only the axioms

of O, but the entire *import closure* of O. The *import closure* of an ontology O is the smallest set containing the axioms of O and of all the axioms of the ontologies that O imports. For example, if ontology O_A imports O_B, and O_B imports O_C, then O_A imports both O_B and O_C.

Implicit Import Declaration The declaration of `owl:imports` primitives is a not a common practice on the Semantic Web. Most published RDF models do not contain explicit `owl:imports` declarations. For example, among the 228 million of documents in Sindice, only 704 thousands are declaring at least one `owl:imports`. Instead, RDF models generally refer to classes and properties of existing ontologies by their URIs. For example, most FOAF profile documents do not explicitly import the FOAF ontology, but instead just refer to classes and properties of the FOAF vocabulary. Following the W3C best practices [18] and Linked Data principles [2], the URIs of the classes and properties defined in an ontology should be resolvable and should provide the machine-processable content of the vocabulary.

In the presence of dereferenceable class or property URIs, we perform what we call an *implicit import*. By dereferencing the URI, we attempt to retrieve a graph containing the description of the ontological entity and to include its content inside the source RDF model. Also the implicit import is considered transitive. For example, if a RDF model refers to an entity E_A from an ontology O_A, and if O_A refers to an entity E_B in an ontology O_B, then the model imports two ontologies O_A and O_B.

Import Lifting Rules Guha's context mechanism defines the *importsFrom* lifting rule (see Section 2.3 for a definition) which corresponds to the inclusion of one context into another. The *owl:imports* primitive and the implicit import declaration are easily mapped to the *importsFrom* rule.

A particular case is when import relations are cyclic. Importing an ontology into itself is considered a null action, so if ontology O_A imports O_B and O_B imports O_A, then the two ontologies are considered to be equivalent [1]. Based on this definition, we extend Guha's definition to allow cycles in a graph of *importsFrom*. We introduce a new symbol eq, and the notation $eq(c_1, c_2)$ states that c_1 is equivalent to c_2, i.e., that the set of propositions true in c_1 is identical to the set of propositions true in c_2.

Definition 1 (Cyclic Import Rule). *Let c_1 and c_2 be two contexts. If c_1 contains the proposition $importsFrom(c_1, c_2)$ and c_2 the proposition $importsFrom(c_2, c_1)$, then the two contexts are considered equivalent:*

$$ist(c_2, importsFrom(c_2, c_1)) \wedge ist(c_1, importsFrom(c_1, c_2)) \rightarrow eq(c_1, c_2)$$

3.2 Deductive Closure of RDF Models

In context-dependent reasoning, the deductive closure of an entity description is the set of assertions that is entailed in the aggregate context composed of the entity description and its ontology import closure. Before defining formally the deductive closure of an aggregate context, we discuss the incomplete reasoning aspect of our approach.

Incomplete Reasoning with Web Data When reasoning with Web data, we can not expect to deal with a level of expressiveness of OWL-DL [5], but would need to consider OWL Full. Since under such circumstances, we can not strive for a complete reasoning anyway, we therefore content ourselves with a finite entailment regime based on a subset of RDFS and OWL, namely the ter Horst fragment [14]. Such a deductive closure provides useful RDF(S) and OWL inferences such as class hierarchy, equalities or property characteristics (inverse functional properties or annotation properties), and is sufficient with respect to increasing the precision and recall of the search engine.

A rule-based inference engine is currently used to compute full materialisation of the entailed statements with respect to this finite entailment regime. In fact, a finite deduction is a requirement in such a setting, which in terms of OWL Full can only be possible if the entailment regime is incomplete (as widely known the RDF container vocabulary alone is infinite already [11]). This is deliberate and we consider a higher level of completeness hardly achievable on the Web of Data: in a setting where the target ontology to be imported may not be accessible at the time of the processing, for instance, we just ignore the imports primitives and are thus anyway incomplete from the start. Also the context-dependent reasoning approach misses inferences from rules that requires multiple A-Box joins across contexts, for example with a transitivity rule across entity descriptions. However, completeness is not our aim. Instead our goal is to take the best out of the data we have and in a very efficient way.

Deductive Closure of Aggregate Context We now explain how the deductive closure of an aggregate context is performed. Given two contexts c_1 and c_2, for example an entity description and an ontology, their axioms are lifted (see Section 2.3 for a definition) into an aggregate context labelled $c_1 \wedge c_2$. The deductive closure of the aggregate context is then computed using the rule-based inference engine.

It is to be noticed that the deductive closure of an aggregate context can lead to inferred statements that are not true in any of the source contexts alone. For example, if a context c_1 contains an instance x of the class *Person*, and a context c_2 contains a proposition stating that *Person* is a subclass of *Human*, then the entailed conclusion that x is a human is only true in the aggregate context $c_1 \wedge c_2$:

$$ist(c_1, Person(x)) \ \wedge \ ist(c_2, subClass(Person, Human)) \\ \rightarrow ist(c_1 \wedge c_2, Human(x))$$

The set of inferred statements that are not true in any of the source contexts alone are called *aggregate entailment*:

Definition 2 (Aggregate Entailment). *Let c_1 and c_2 be two contexts with respectively two propositions φ_1 and φ_2, $ist(c_1, \varphi_1)$ and $ist(c_2, \varphi_2)$, and $\varphi_1 \wedge \varphi_2 \models \varphi_3$, such that $\varphi_2 \not\models \varphi_3$, $\varphi_1 \not\models \varphi_3$, then we call φ_3 a newly entailed proposition in the aggregate context $c_1 \wedge c_2$. We call the set of all newly defined propositions an aggregate entailment and denote it as Δ_{c_1,c_2}:*

$$\Delta_{c_1,c_2} = \{ist(c_1, \varphi_1) \wedge ist(c_2, \varphi_2) \models ist(c_1 \wedge c_2, \varphi_3) \\ \textit{and } \neg(ist(c_1, \varphi_3) \vee ist(c_2, \varphi_3))\}$$

The aggregate entailment property enables the reasoning engine to confine inference results to specific contexts and therefore protects other contexts from unexpected data usage. Unexpected data usage in one context will not alter the intended semantics of other contexts, if and only if no direct or indirect import relation exists between them.

Note that - by considering (in our case (Horn) rule-based) RDFS/OWL inferences only - aggregate contexts enjoy the following monotonicity property[3]: if the aggregate context $c_1 \subseteq c_2$ then $ist(c_2, \phi)$ implies $ist(c_1, \phi)$, or respectively, for overlapping contexts, if $ist(c_1 \cap c_2, \phi)$ implies both $ist(c_1, \phi)$ and $ist(c_2, \phi)$. This property is exploited in our ontology base, which is described next, to avoid storing duplicate inferred statements.

4 Context-Dependent Ontology Base

A problem when reasoning over a large number of entity descriptions independently is that the process of computing the ontology import closure and its deductive closure has to be repeated for each entity description. This is inefficient since the computation of the import closure and the deductive closure is resource demanding and can in fact be reused for other entity descriptions. The import closure necessitates to execute multiple web requests that are network resource demanding and time consuming, while the computation of the deductive closure is CPU bound. In addition, we observe in Section 6 that the computation of the T-Box closure is more CPU intensive than the computation of the A-Box closure. This observation suggests to focus on the optimisation of the T-Box closure computation. Thanks to the smaller scale of the T-Box with respect to the A-Box, we can store the computed ontology import closure as well as the deductive closure in a *ontology base* in order to reuse them in later computation.

The ontology base, which can be seen as a persistent context-dependent T-Box, is in charge of storing any ontology discovered on the web along with their import relations. The ontology base also stores the inference results that has been performed in order to reuse them later. The ontology base serves the inference engine by providing the appropriate and pre-computed T-Box for reasoning over an entity description.

In the next, we first introduce the basic concepts used in the formalisation of the ontology base. We then discuss an optimised strategy to update the ontology base. We finally describe how to query the ontology base.

4.1 Ontology Base Concepts

The ontology base uses the notion of named graphs [4] for modelling the ontologies and their import relations. The ontology base relies on a rules-based inference engine to compute the deductive closure of either one ontology or of a combination of them.

Ontology Entity An *Ontology Entity* is a property, instance of `rdf:Property`, or a class, instance of `rdfs:Class`. The ontology entity must be identified by an URI (we exclude entities that are identified by a blank node) and the URI must be resolvable and must point to a document containing the ontological entity description.

[3] We remark here that under the addition of possibly non-monotonic rules to the Semantic Web architecture, this context monotonicity only holds under certain circumstances [19].

Ontology Context An *Ontology Context* is a named graph composed by the ontology statements that have been retrieved after dereferencing the entity URIs of the ontology. The content of this graph consists of the union of the descriptions of all properties and classes associated to a same ontology namespace. According to best practices [18], properties and classes defined in an ontology should have the same URI namespace.

Usually, this means that the data associated with the ontology context simply reflects the content of the ontology document as found on the Web. There are cases however, e.g., in the case of the OpenCyc ontology[4], where each property and class has its own view. The ontology context is therefore composed by the union of all these views.

Ontology Network An ontology context can have import relationships with other ontology contexts. A directed link l_{AB} between two contexts, O_A and O_B, stands for O_A imports O_B. A link l_{AB} is mapped to an *importsFrom* lifting rule and serves as a pointer to a frame of knowledge that is necessary for completing the semantics of O_A.

Definition 3 (Ontology Network). *An ontology network is a directed graph $\langle \mathcal{O}, \mathcal{L} \rangle$ with $\mathcal{O}$ a set of vertices (ontology contexts) and $\mathcal{L}$ a set of directed edges (import relations).*

The import closure of an ontology can be seen as an activation of a subset of the ontology network.

Definition 4 (Import Closure). *The import closure of an ontology context O_A for an ontology network $\langle \mathcal{O}, \mathcal{L} \rangle$ is a subgraph $\langle \mathcal{O}', \mathcal{L}' \rangle$ such that:*

$$\mathcal{O}' \subseteq \mathcal{O}, \mathcal{L}' \subseteq \mathcal{L} \mid \forall O_i \in \mathcal{O}', \exists\, path(O_A, O_i)$$

where $path(O_A, O)$ denotes a sequence of vertices $O_A, \ldots, O_i$ such that from each of its vertices there is an edge to the next vertex.

For the purpose of this work, we consider the import closure to also contain the deductive closure of the union of all the ontologies. For example, given two ontology contexts O_A and O_B, with O_A importing O_B, the import closure of O_A will consist of the context aggregating O_A, O_B and the deductive closure of the union of the two ontologies (i.e., the entailment of O_A and O_B as well as the *aggregate entailment* Δ_{O_A,O_B}).

4.2 Ontology Base Update

When a new ontology context is added to the ontology network, its import closure is materialised. Then, the deductive closure is computed by first lifting the axioms of the import closure and by computing the newly aggregate entailment as explained in Definition 2. Finally the new statements are added to the ontology network so that they are never duplicated.

This is better explained by an example. In Figure 1a, an entity E_1 imports explicitly two ontologies O_1 and O_2. The import closure of E_1 is calculated and this is found to include also O_3 since O_3 is imported by O_2. At this point, the following step are performed:

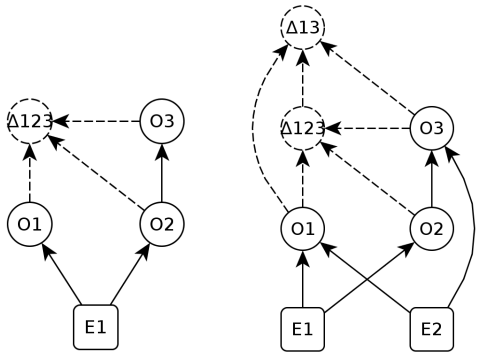
[4] OpenCyc: http://sw.opencyc.org/

1. The deductive closure of O_1 and O_3 is computed separately and stored in their respective nodes.
2. The deductive closure of O_2 is computed by first lifting the axioms of O_3. The entailed propositions Δ_{O_2,O_3} are stored in O_2.
3. Finally, the deductive closure of O_1, O_2 and O_3 is calculated. The entailed statements Δ_{O_1,O_2,O_3} resulting from the reasoning over O_1, O_2 and O_3 together but that are not found already in any of the source contexts are stored in a virtual context Δ_{123}.

At this point a new entity E_2 comes which only imports O_1 and O_3 as shown in Figure 1b. The update strategy will:

1. calculate the deductive closure of O_1 and O_3 and store the new assertions Δ_{O_1,O_3} in a new virtual context Δ_{13};
2. subtract these triples from the content of the previous context Δ_{123}. Δ_{123} is connected to Δ_{13} by an import relation.

A last optimisation is based on Definition 1. Whenever a cycle is detected into the ontology network, the ontology contexts present in the cycle are aggregated into one unique context.



(a) The ontology network after processing an entity E_1.

(b) The ontology network after processing a second entity E_2.

Fig. 1: Example of ontology network

4.3 Ontology Base Querying

The ontology base is used by the A-Box reasoner to retrieve the appropriate T-Box for each entity description being processed. The A-Box reasoner first performs an analysis of the entity description and extracts all the references to the ontological terms being used. It then queries the ontology base with the set of ontological terms.

The ontology base maps each ontological terms to their ontology and computes the import closure for each of the ontology. Finally, it merges the different import closures,

including the aggregate entailments Δ, into one model and returns this model to the A-Box reasoner to be used as T-Box.

For example in Figure 1b, the entity E_1 imports the two ontologies O_1 and O_2. This lifts the contexts $\{O_1, O_2, O_3, \Delta_{13}, \Delta_{123}\}$ into the entity E_1 before computing the A-Box deductive closure.

Sometimes, the computed T-Box can be large for two reasons: (1) when the import closure is large, i.e., a large number of ontologies are merged together; or (2) when large ontologies are present in the import closure. As a consequence, the amount of information transferred through the network between the A-Box reasoner and the ontology base increases. In addition, the memory requirement of the A-Box reasoner also increases. However, we found that the A-Box reasoner does not need the complete T-Box, i.e., the complete set of ontologies, but that only a subset of the T-Box could be used instead without having any impact on the A-Box inference result. This T-Box subset is composed of the union of the descriptions of all the ontological entities used in the original entity description. To extract the description of one ontological entity, the ontology base relies on a technique similar to the Symmetric Concise Bounded Description [21] where not only blank nodes are followed, but also the symmetric property `owl:inverseOf`. [1] states that "an axiom of the form `P1 owl:inverseOf P2` asserts that for every pair (x,y) in the property extension of P1, there is a pair (y,x) in the property extension of P2, and vice versa". By not including the description of the inverse property, the A-Box reasoner might miss some logical assertions about the property.

5 Implementation

The ontology base is implemented using a RDF database to store the ontology statements in their context. A secondary index is used to store the import relations between the contexts. A caching mechanism is used on top of the ontology base to cache frequent requests. The caching mechanism is especially useful when processing multiple entities from a single dataset. Since entities from a same dataset are likely to be described with the same ontologies, the requests to the ontology base are identical and the cache hit rate increases.

The reasoning engine which is used by the ontology base is especially designed and optimised to compute entailments in memory. Each term (i.e., URIs, Blank Nodes and Literals) in a statement is mapped to a unique identifier (integer). Statements are indexed using in-memory data structures, similar to triple tables, in order to lookup any kind of statement patterns. Rules are then checked against the index in an iterative manner, with one rule being applied at a given iteration. The result of the rule is then added to the index before proceeding to the next iteration. Iterations continue until a fixpoint is reached. For rules that requires joins between multiple statements, since we are working with a small amount of data and a small number of elements, we rely on an efficient merge-join algorithm where both relations are sorted on the join attribute using bit arrays. The bit arrays are then intersected using bitwise operations.

The A-Box reasoning is distributed on a cluster of machines using the following strategy. In our approach, the A-Box is divided on a per-context basis (in our settings, on a per-entity basis). Each context provides a chunk of data which is distributed to

different computing nodes. A computing node acts as A-Box reasoner and has its own ontology base. The A-Box rule engine is based on the same rule engine used by the ontology base.

Since each chunk of data is relatively small, the deductive closure of the A-Box can be entirely performed in memory without relying on disk accesses. With respect to other distributed approaches that perform reasoning on the global model, we avoid to read and write multiple times the data directly from the disk, and therefore we obtain better performance. Finally, it is to be noticed that such a distributed model scales linearly with the number of available nodes in the cluster.

6 Performance Evaluation

In this section, we evaluate the performance of our approach on three datasets:

Geonames: is a geographical database and contains 13.8 million of entities[5].
DBPedia: is a semi-structured version of Wikipedia and contains 17.7 million of entities[6].
Sindice: is a representative subset of the Web of Data containing, at the time of the experiment, more than 110 million of documents. It is composed of Semantic Web online repositories and pages with Microformats or RDFa markups crawled on a regular basis for more than three years.

We first start by comparing the size of the T-Box and A-Box data for each dataset. We then benchmark the performance of the context-dependent reasoning approach with and without an ontology base. We finally estimate the performance in a distributed setting.

6.1 Quantifying T-Box and A-Box

We perform a random sample of 100.000 entities for each dataset. For each entity, we apply our context-dependent reasoning approach and record various characteristics such as:

Ontology: the number of ontologies in the ontology closure;
T-Box - Explicit: the number of axioms in the original ontology closure;
T-Box - Implicit: the number of entailed statements from the ontology closure;
A-Box - Explicit: the number of facts in the original entity description;
A-Box - Implicit: the number of inferred facts from the entity description and the ontology closure.

Table 1 reports the arithmetic mean of the 100.000 measurements for each dataset.

We can notice that in general the ontology import closure is relatively large with in average 14 and 15 ontologies per entity on the Sindice and Geonames dataset respectively. The DBpedia dataset has in average less ontologies. This can be explained by the fact that this dataset contains many small entity descriptions which solely consist of a

[5] Geonames: http://www.geonames.org/
[6] DBpedia: http://dbpedia.org/

		T-Box Size		A-Box Size	
Dataset	Ontology	Explicit	Implicit	Explicit	Implicit
Geonames	15	1820	4005	6	14
DBPedia	8	657	1556	7	16
Sindice	14	2085	4601	69	170

Table 1: Statistics about T-Box and A-Box of 100.000 random entity descriptions.

few links of type *dbpedia:wikilink* with other entities. However, we report that it also contains some fairly complex ones having an import closure of more than 30 ontologies.

These ontologies account for a large number of axioms, up to 2085 for Sindice dataset. The deductive closure provides a larger number of additional axioms. On all the datasets, the ratio of inferred statements is around 2.2, i.e., 2.2 implicit axioms in average are inferred for 1 explicit axiom.

The A-Box is relatively small compared to the T-Box. The ratio between T-Box and A-Box statements varies greatly across datasets, with 300:1 for Geonames, 94:1 for DBPedia and 30:1 for Sindice. In average, the ratio of inferred statements on the A-Box varies between 2 and 3, which is very close to the inference ratio of the T-Box.

These statistics show that, when reasoning over an entity, there is more data to be processed in the T-Box than there is in the A-Box, suggesting that most of the computational complexity will reside on the T-Box level. Therefore, these observations strengthen the needs of an appropriate caching system such as the ontology base to reuse T-Box computation whenever it is possible.

6.2 Optimisation with Ontology Base

In the next experiment, we show the performance improvements provided by the ontology base. We compare the time to reason over an entity with and without ontology base. We then show the gradual performance improvement as the ontology base is being updated.

Experimental Settings The hardware system we use in our experiments is a 2 x Opteron 250 @ 2.4 GHz (2 cores, 1024 KB of cache size each) with 4GB memory and a local SATA disk. The operating system is a 64-bit Linux 2.6.31-20-server. The version of the Java Virtual Machine (JVM) used during our benchmarks is 1.6.0_20. The reasoning engine and the ontology base is written in Java. The A-Box reasoner runs on a single thread.

Experimental Design In the first experiment, we reason sequentially over 100.000 randomly selected entities with and without the ontology base activated. When the ontology base is deactivated, we measure the time to compute the deductive closure of the T-Box and A-Box for each entity. When the ontology base is activated, we measure the time to request the T-Box from the ontology base and to perform the deductive closure of the A-Box. Beforehand, we load all the ontology closures, i.e., their import closures and their deductive closures, in the ontology base.

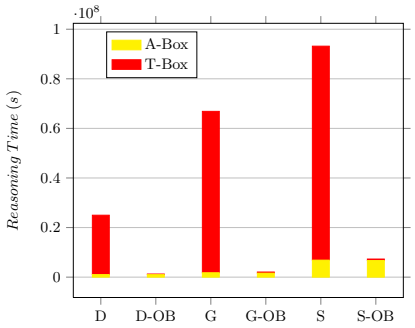
In the second experiment, we start with an empty ontology base and record the evolution of the computational time while the ontology base is being created,i.e., while the import and deductive closures are being computed and stored.

In the two experiments, we ignore the time spent by the ontology base to resolve URIs and fetch the associated documents from the Web. This is deliberate since such a time is dependent of the network resources available and of the availability of the documents on the Web at the time of the fetching.

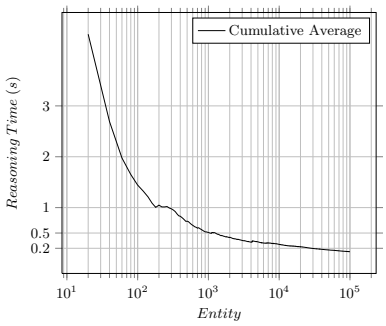
Experimental Results Figure 2a reports the total time to reason over the 100.000 entities for each dataset. The total time is composed of the time spent computing the A-Box and T-Box closure. The bar *D*, *G* and *S* denotes the time to reason without an ontology base on DBpedia, Geonames and Sindice dataset respectively. The bar *D-OB*, *G-OB* and *S-OB* denotes the reasoning time with an ontology base. Each bar is divided in two parts in order to differentiate the time spent to compute the A-Box and T-Box closures.

A first observation is that the time increases with the complexity of the dataset and with the number of statements inferred. DBpedia is the fastest to process following by Geonames and then Sindice. We can also notice that the time to compute the closure of the A-Box on Sindice is much higher than on the other datasets, which correlates with the larger number of A-Box statement inferred found in Table 1.

The most interesting result is the improvement provided by the ontology base. On the three datasets, we can observe that the computation of the T-Box closure is the most time consuming operation. However, it becomes negligible when the ontology base is activated.



(a) The total time in second to reason over 100.000 randomly selected entities.



(b) The cumulative average of the reasoning time per entity in second over a sequence of 100.000 randomly selected entities from the Sindice dataset.

Figure 2b depicts the performance improvement of the reasoning engine while the ontology base is being updated, i.e., when the import and deductive closure are being computed for each new ontology. We analyse the measurements by computing the cumulative average of the reasoning time over the past five entities. We can see that at the beginning, while the ontology base is still empty, the reasoning time is very high with more than 3 seconds. However, the time rapidly decreases under the second and starts to reach its steady state of performance after 10000 entities.

6.3 Distributed Performance

Since the distributed model scales linearly with the number of nodes available in the cluster, we can estimate the number of entities per second that can be processed on a given number of nodes. This estimation is based on the previous results where in average 83.3, 50 and 13.9 entities per second where processed on DBpedia, Geonames and Sindice respectively. A system with 32 nodes is able to ingest more than 2000 entities per second on DBPedia, more than 1500 on Geonames and more than 400 on heterogeneous data collection such as Sindice. It is to be noticed that these results come from a prototypical implementation, still to be subject to many possible technical optimisations.

7 Discussion and Future Works

The technique presented in this chapter is mainly focussed on the T-Box level. The import relations between ontologies provide a good support for lifting rules. However, on A-Box level, it is not clear which relations between entity descriptions should be consider as lifting rules. We will investigate equality relations such as the `owl:sameAs` relations in the future. But even more importantly there is a concept of "dataset scope" when applying said rules. For example, these rules might be applied differently within a dataset or across datasets[7] as the level of trust is different. This calls for augmentation of this work and will be discussed in future works.

8 Conclusions

We report our experiences and lessons learned in building a context dependent methodology for large scale web data reasoning. We first define the problem conceptually and then illustrate how to create an ontology repository which can provide the materialization of implicit statements while keeping track of their provenance. Our implementation currently support a subset of RDFS and OWL. We find this level of inference to be in line with Sindice's target objective to support the RDF community of practice, e.g., the Linked Data community, which usually relies only on RDFS and OWL features covered by the OWL ter Horst fragment [14]. The reasoning methodology has been deployed, used and improved since 2008 within Sindice and has enabled Sindice to reason over billions of triples.

The context mechanism allows Sindice to avoid the deduction of undesirable assertions in RDF models, a common risk when working with the Web Data. This context mechanism does not restrict the freedom of expression of data publishers. Data publishers are still allowed to reuse and extend ontologies in any manner, but the consequences of their modifications will be confined in their own context, and will not alter the intended semantics of the other RDF models published on the Web.

[7] A dataset denotes a web site with metadata markups or a RDF database.

References

1. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference. W3C Recommendation, W3C (February 2004)
2. Berners-Lee, T.: Linked data. W3C Design Issues (July 2006), http://www.w3.org/DesignIssues/LinkedData.html
3. Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., Stuckenschmidt, H.: Contextualizing ontologies. Journal of Web Semantics 1(4), 325–343 (2004)
4. Carroll, J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, provenance and trust. In: WWW '05: Proceedings of the 14th international conference on World Wide Web. pp. 613–622. ACM, New York, NY, USA (2005)
5. d'Aquin, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Sabou, M., Motta, E.: Characterizing Knowledge on the Semantic Web with Watson. In: EON. pp. 1–10 (2007)
6. Delbru, R., Polleres, A., Tummarello, G., Decker, S.: Context Dependent Reasoning for Semantic Documents in Sindice. In: Proceedings of the 4th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2008) (2008)
7. Ding, L., Tao, J., McGuinness, D.L.: An initial investigation on evaluating semantic web instance data. In: WWW '08: Proceeding of the 17th international conference on World Wide Web. pp. 1179–1180. ACM, New York, NY, USA (2008)
8. Giunchiglia, F.: Contextual reasoning. Epistemologia, special issue on I Linguaggi e le Macchine 345, 345–364 (1993)
9. Guha, R.V.: Contexts: a formalization and some applications. Ph.D. thesis, Stanford, CA, USA (1992)
10. Guha, R.V., McCool, R., Fikes, R.: Contexts for the Semantic Web. In: International Semantic Web Conference. pp. 32–46 (2004)
11. Hayes, P.: RDF Semantics. W3C Recommendation, W3C (February 2004)
12. Hogan, A., Harth, A., Polleres, A.: Scalable Authoritative OWL Reasoning for the Web. International Journal on Semantic Web and Information Systems 5(2), 49–90 (2009)
13. Hogan, A., Pan, J.Z., Polleres, A., Decker, S.: SAOR: Template Rule Optimisations for Distributed Reasoning over 1 Billion Linked Data Triples. In: Proceedings of the 9th International Semantic Web Conference. Springer (2010)
14. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. Journal of Web Semantics 3(2-3), 79–115 (2005)
15. de Kleer, J.: An Assumption-Based TMS. Artif. Intell. 28(2), 127–162 (1986)
16. Mayfield, J., Finin, T.: Information retrieval on the Semantic Web: Integrating inference and retrieval. In: Proceedings of the SIGIR Workshop on the Semantic Web (August 2003)
17. McCarthy, J.: Notes On Formalizing Context. In: Proceedings of IJCAI-93. pp. 555–560 (1993)
18. Miles, A., Baker, T., Swick, R.: Best Practice Recipes for Publishing RDF Vocabularies. W3C working group note, W3C (2008), http://www.w3.org/TR/swbp-vocab-pub/
19. Polleres, A., Feier, C., Harth, A.: Rules with contextually scoped negation. In: 3rd European Semantic Web Conference (ESWC2006). LNCS, vol. 4011. Springer, Budva, Montenegro (Jun 2006), http://www.polleres.net/publications/poll-etal-2006b.pdf
20. Serafini, L., Bouquet, P.: Comparing formal theories of context in AI. Artificial Intelligence 155(1-2), 41 (2004)
21. Stickler, P.: CBD - Concise Bounded Description. W3C Member Submission, W3C (June 2005)

22. Stoermer, H., Bouquet, P., Palmisano, I., Redavid, D.: A Context-Based Architecture for RDF Knowledge Bases: Approach, Implementation and Preliminary Results. In: RR. pp. 209–218 (2007)
23. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.: OWL Reasoning with WebPIE: Calculating the Closure of 100 Billion Triples. In: Aroyo, L., Antoniou, G., Hyvönen, E., Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) Proceedings of the 7th Extended Semantic Web Conference, ESWC2010. Lecture Notes in Computer Science, vol. 6088, pp. 213–227. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
24. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable distributed reasoning using mapreduce. In: International Semantic Web Conference. pp. 634–649 (2009)
25. Weaver, J., Hendler, J.A.: Parallel materialization of the finite rdfs closure for hundreds of millions of triples. In: International Semantic Web Conference (ISWC2009). pp. 682–697 (2009)