SAOR: Authoritative Reasoning for the Web*

Aidan Hogan and Andreas Harth and Axel Polleres

Digital Enterprise Research Institute, National University of Ireland, Galway

Abstract. In this paper we discuss the challenges of performing reasoning on large scale RDF datasets from the Web. We discuss issues and practical solutions relating to reasoning over web data using a rule-based approach to forward-chaining; in particular, we identify the problem of ontology hijacking: new ontologies published on the Web re-defining the semantics of existing concepts resident in other ontologies. Our solution introduces consideration of authoritative sources. Our system is designed to scale, comprising of file-scans and selected lightweight on-disk indices. We evaluate our methods on a dataset in the order of a hundred million statements collected from real-world Web sources.

1 Introduction

Data attainable through the Web is unique in terms of scale and diversity. Millions of data sources contribute billions of statements to a giant data graph. The Semantic Web technology stack includes means to supplement instance data being published using the Resource Description Framework (RDF) with ontologies described in RDF Schema (RDFS) [1] and the Web Ontology Language (OWL) [16], allowing people to formally specify a domain of discourse, and providing machines a more sapient understanding of the data. While there exists a large body of work in the area of reasoning algorithms and systems that work and scale well in confined environments, the distributed and loosely coordinated creation of a world-wide knowledge base creates new challenges.

Reasoning over aggregated Web data is useful, for example: to infer new assertions using terminological knowledge from ontologies and therefore provide a more complete dataset; to unite fractured knowledge about individuals collected from disparate sources; and to execute mappings between domain descriptions and therefore provide translations from one conceptual model to another. Our work on reasoning is motivated by the requirements of the Semantic Web Search Engine (SWSE) project[1], within which we strive to offer search, querying and browsing over the Semantic Web.

Reasoning on Web data poses a number of requirements:

- the system has to perform on web-scale, with implications on the completeness of the reasoning procedure, algorithms and optimisations

* This work has been supported by Science Foundation Ireland (SFI/02/CE1/I131), European FP6 project inContext (IST-034718) and COST Action "Agreement Technologies" (IC0801).

[1] http://swse.deri.org/

– the method has to perform on collaboratively created knowledge bases, which has implications on trust and the privileges of data publishers
– the web search scenario requires sub-second response times, which has implications on the reasoning and query processing strategy

We present SAOR – Scalable Authoritative OWL Reasoner – which focuses on performing best-effort RDFS and OWL reasoning on Web data. SAOR is designed to accept as input a web knowledge-base in the form of a body of statements as produced by a web-crawl and to output by forward-chaining a knowledge-base enhanced by a given fragment of OWL reasoning. Discussion of the end consumption of such a reasoned knowledge-base is outside of the scope of this paper.

Specifically, we make the following contributions in this paper:

– We apply only a selected subset of OWL reasoning, to i) avoid an explosion of inferred statements and ii) to protect existing specifications from undesirable contributions made in independent locations. Our system implements a positive fragment of OWL Full which has roots in ter Horst's *pD** [17] entailment. We describe an analysis of the authority of sources to counter-act the highlighted problem of *ontology hijacking* in web data (Section 2).
– We describe a scalable, optimised method for performing rule-based forward chaining reasoning through means of file-scan and lightweight dynamic data structures. In particular, our algorithm capitalises on the low volume of structural T-Box data relative to A-Box instance data (Section 3).
– We show experimentally that a forward-chaining materialisation approach is feasible on Web data from 315k sources and in the order of 100m triples: with our confined reasoning strategy we found that the knowledge base only roughly doubles in size (Section 4) by cautious materialisation.

We discuss related work in Section 5 and conclude with Section 6.

2 Pragmatic Inferencing for the Web

We begin by stating a couple of observations regarding the feasibility of reasoning on the Web. Firstly, most OWL data crawlable on the Web is OWL Full. Idealised assumptions made in OWL DL, such as disallowing subclassing or defining subproperties of the OWL and RDF(S) vocabularies, are violated by even very commonly used ontologies[2]. Secondly, consistency cannot be expected on the Web. For instance, a past web-crawl of ours revealed the following:[3]

```
<timbl> a foaf:Person; foaf:homepage <http://w3.org/> .
<w3c> a foaf:Organization; foaf:homepage <http://w3.org/> .
foaf:homepage a :InverseFunctionalProperty.
foaf:Organization :disjointWith foaf:Person .
```

[2] E.g., one of the reasons why the commonly used FOAF vocabulary falls into OWL Full is that `foaf:name` is defined as a subproperty of `rdfs:label` [19].

[3] Throughout this paper, we assume that `http://www.w3.org/2002/07/owl#` is the default namespace and use well understood prefixes for other namespaces.

These triples together infer that Tim Berners-Lee is the same as the W3C and cause an inconsistency. However, despite such examples which arise from misunderstanding of the FOAF vocabulary, there might be cases where different parties deliberately make contradictive statements.

These two points already suggest that complete inference at the instance level is neither feasible nor desirable: firstly, for the computational infeasibility of complete OWL Full reasoning, and secondly, since we do not deem the explosive nature of contradiction in classical logics desirable in a Web reasoning scenario.

Thus, rather than striving for complete inference, we adopt a "best effort" reasoning strategy, optimising inference based on the following principles:

1. We assume a separation of T-Box from A-Box.
2. We trade completeness for implementational feasibility following a rule-based, finite, forward-chaining approach to OWL inference.
3. We trade completeness for producing a much smaller subset of inferred statements; i.e, we deliberately ignore (i) the explosive behaviour of classical inconsistency, (ii) arguably "void" statements in terms of non-standard use of the RDF(S) and OWL vocabularies, (iii) *non-authoritative* T-Box statements.

2.1 Separating A-Box from T-Box

In SAOR, we separate terminological knowledge from assertional data according to their use of the RDF(S) and OWL vocabulary; we call these the "A-Box" and the "T-Box" respectively (loosely borrowing Description Logics terminology).

Table 1 provides a list of graph patterns in RDF graphs we consider to be part of the T-Box. Note that when retrieving graphs from the Web, the instances of these patterns are all of the T-Box statements we consider in our reasoning process: triples that do not match one of these patterns are not considered being part of the T-Box, but are treated purely as assertional "data" triples.

The materialisation of axiomatic statements and completing the entire T-Box may create a bulk of statements with little practical utility. In fact, we deliberately accept the omission of T-Box inference rather as an optimisation: we focus on answering queries over A-Box data rather than, e.g., inferring all members of `:Class`.

SAOR does not support metamodelling, except by conceptually separating the instance- class- or property-meanings of a resource: by separating the T-Box and A-Box segment of the knowledge base, we do not support all possible entailments from the simultaneous description of both a class and an instance. Particularly, we treat URIs in the context they appear, in the spirit of "punning"[4]. We do subject the T-Box data to reasoning analogously to the A-Box, but only store results in the A-Box. For example, we do not carry over `:sameAs` inferences to the T-Box – this is in-line with first-order-logic point of view, where equalities do not affect predicates.

We filter out further triples when extracting the T-Box; namely, we ignore nonstandard use of RDF in our reasoning efforts. Non-standard use of RDF

[4] http://www.w3.org/2007/OWL/wiki/Punning

No	DL Syntax	Corresponding RDF graph pattern
01	$C \sqsubseteq D$	**?C** rdfs:subClassOf ?D .
02_a	$C \equiv D$	**?C** :equivalentClass ?D .
02_b		?C :equivalentClass **?D** .
03	$P \sqsubseteq Q$	**?P** rdfs:subPropertyOf ?Q .
04_a	$P \equiv Q$	**?P** :equivalentProperty ?Q .
04_b		?P :equivalentProperty **?Q** .
05_a	$P \equiv Q^-$	**?P** :inverseOf ?Q .
05_b		?P :inverseOf **?Q** .
06	$\top \sqsubseteq \forall P^-.C$	**?P** rdfs:domain ?C .
07	$\top \sqsubseteq \forall P.C$	**?P** rdfs:range ?C .
08	$P \equiv P^-$	**?P** a :SymmetricProperty .
09_a	$\exists P.x$	?C :hasValue ?x; :onProperty **?P** .
09_b		**?C** :hasValue ?x; :onProperty ?P .
10	$C_1 \sqcup ... \sqcup C_n$	?C :unionOf (?C_1 ... **?C_i** ... ?C_n) .
11_a	$C_1 \sqcap ... \sqcap C_n$	**?C** :intersectionOf (?C_1 ... ?C_n) .
11_b		?C :intersectionOf (*?C_1* ... *?C_n*) .
12	$\top \sqsubseteq \forall \leq 1P$	**?P** a :FunctionalProperty .
13	$\top \sqsubseteq \forall \leq 1P^-$	**?P** a :InverseFunctionalProperty .
14	$P^+ \sqsubseteq P$	**?P** a :TransitiveProperty .
15	$\exists P.D$	?C :someValuesFrom *?D*; :onProperty *?P* .
16	$\forall P.D$	*?C* :allValuesFrom ?D; :onProperty *?P* .
17_a	$(\leq 1P)$	*?C* :maxCardinality 1; :onProperty *?P* .
17_b	$(= 1P)$	*?C* :cardinality 1; :onProperty *?P* .
18	$\{x_i....x_n\}$	?C :oneOf (?x1 ... ?xN) .

Table 1: Allowed T-Box constructs for each rule. Bold type indicates that the element must be authoritatively spoken for. Where they appear, at least one of the italic type elements must be authoritatively spoken for (see 2.3).

briefly equates to the use of properties and classes which make up the RDF(S) vocabulary in locations where they have not been intended, cf. [2, 13]. We adapt the definition of non-standard use for our purposes and only restrict the usage of the vocabulary for the T-Box: let $\mathcal{P} = \{$ `rdf:type, rdf:domain, rdf:range, rdfs:subClassOf, rdfs:subPropertyOf, :equivalentClass, :equivalentProperty, :inverseOf, :onProperty, :hasValue, :someValuesFrom, :allValuesFrom, :intersectionOf, :unionOf, :maxCardinality, :cardinality, :oneOf` $\}$, and $\mathcal{C} = \{$ `:FunctionalProperty, :InverseFunctionalProperty, :TransitiveProperty, :SymmetricProperty` $\}$. We omit from the T-Box any triples with non-standard use of the properties and classes in $\mathcal{P} \cup \mathcal{C}$, that is, triples where properties in $\mathcal{P}$ appear in a position other than the predicate position or where classes in $\mathcal{C}$ appear in a position other than the object of an `rdf:type` triple.

In summary, our view of a web knowledge base $\mathcal{KB}$ consists of the RDF merge of a set of source graphs. $\mathcal{KB}$ is separated into a T-Box $\mathcal{T}$ mentioning classes and properties $\mathcal{C}_{\mathcal{KB}}$ and $\mathcal{P}_{\mathcal{KB}}$ and an A-Box $\mathcal{A}$ consisting of class and property membership assertions possibly using identifiers in $\mathcal{C}_{\mathcal{KB}} \cup \mathcal{P}_{\mathcal{KB}}$.

2.2 Rule-based OWL Reasoning

Reasoning in SAOR is inspired by previous approaches, particularly the *pD** fragment defined by ter Horst [17], to cover large parts of OWL by positive inference rules which can be implemented in a forward-chaining engine.

Table 2 lists all of the currently supported rules. Although certain triples matched in the antecedents come from the T-Box and others come from the A-Box, in contrast to ter Horst's original rules, inferences are stored in the A-Box only. Thus, on exhaustive application of the rules, the T-Box remains unchanged.

	DL Syntax	Rule
01	$C \sqsubseteq D$	*?C rdfs:subClassOf ?D* . ?s a ?C . ⇒ ?s a ?D .
02_a	$C \equiv D$	*?C :equivalentClass ?D* . ?s a ?C .⇒ ?s a ?D .
02_b		*?C :equivalentClass ?D* . ?s a ?D .⇒ ?s a ?C .
03	$P \sqsubseteq Q$	*?P rdfs:subPropertyOf ?Q* . ?s ?P ?o . ⇒ ?s ?Q ?o .
04_a	$P \equiv Q$	*?P :equivalentProperty ?Q* . ?s ?P ?o . ⇒ ?s ?Q ?o .
04_b		*?P :equivalentProperty ?Q* . ?s ?Q ?o . ⇒ ?s ?P ?o .
05_a	$P \equiv Q^-$	*?P :inverseOf ?Q* . ?s ?P ?o . ⇒ ?o ?Q ?s .
05_b		*?P :inverseOf ?Q* . ?s ?Q ?o . ⇒ ?o ?P ?s .
06	$\top \sqsubseteq \forall P^-.C$	*?P rdfs:domain ?C* . ?s ?P ?o . ⇒ ?s a ?C .
07	$\top \sqsubseteq \forall P.C$	*?P rdfs:range ?C* . ?s ?P ?o . ⇒ ?o a ?C .
08	$P \equiv P^-$	*?P a :SymmetricProperty* . ?s ?P ?o . ⇒ ?o ?P ?s .
09_a	$\exists P.x$	*?C :hasValue ?x; :onProperty ?P* . ?y ?P ?x . ⇒ ?y a ?C .
09_b		*?C :hasValue ?x; :onProperty ?P* . ?y a ?C . ⇒ ?y ?P ?x .
10	$C_1 \sqcup ... \sqcup C_n$	*?C :unionOf (?C₁ ... ?Cᵢ ... ?Cₙ)* . ?x a ?Cᵢ[5] . ⇒ ?x a ?C .
11_a	$C_1 \sqcap ... \sqcap C_n$	*?C :intersectionOf (?C₁ ... ?Cₙ)* . ?y a ?C . ⇒ ?y a ?C₁, ..., ?Cₙ .
11_b		*?C :intersectionOf (?C₁ ... ?Cₙ)* . ?y a ?C₁, ..., ?Cₙ . ⇒ ?y a ?C .
12	$\top \sqsubseteq \forall \leq 1P$	*?P a :FunctionalProperty* . ?s ?P ?x , ?y . ⇒ ?x :sameAs ?y .
13	$\top \sqsubseteq \forall \leq 1P^-$	*?P a :InverseFunctionalProperty* . ?x ?P ?o . ?y ?P ?o . ⇒ ?x :sameAs ?y .
14	$P^+ \sqsubseteq P$	*?P a :TransitiveProperty* . ?x ?P ?y . ?y ?P ?z . ⇒ ?x ?P ?z .
15	$\exists P.D$	*?C :someValuesFrom ?D; :onProperty ?P* . ?x ?P ?y . ?y a ?D . ⇒ ?x a ?C .
16	$\forall P.D$	*?C :allValuesFrom ?D; :onProperty ?P* . ?x ?P ?y; a ?C . ⇒ ?y a ?D .
17_a	$(\leq 1P)$	*?C :maxCardinality 1; :onProperty ?P* . ?x a ?C; ?P ?y, ?z . ⇒ ?y :sameAs ?z .
17_b	$(= 1P)$	*?C :cardinality 1; :onProperty ?P* . ?x a ?C; ?P ?y, ?z . ⇒ ?y :sameAs ?z .
18	$\{o_i....o_n\}$	*?C :oneOf (?o₁ ... ?oₙ)* . ⇒ ?o₁ ... ?oₙ a ?C .
19_a	$x = y$	?x :sameAs ?y . ?x ?p ?o .⇒ ?y ?p ?o .
19_b		?x :sameAs ?y . ?s ?p ?x .⇒ ?s ?p ?y .
19_c		?x :sameAs ?y . ⇒ ?y :sameAs ?x .
19_d		?x :sameAs ?y . ?y :sameAs ?z . ⇒ ?x :sameAs ?z .

Table 2: Supported rules with N3-style syntax used for triple patterns with T-Box statements italicised and A-Box in plain font.

Next, we only support inferences in one direction for `:someValuesFrom` and `:allValuesFrom`, as we do not apply any inference rules that involve invention of new blank nodes. Like ter Horst, we do not support inequalities or disjointness; i.e., SAOR operates monotonically without "explosive" reaction in inconsistency. In addition to *pD** we support functional cardinality constraints, as well as limited support for enumerated classes (`:oneOf`).

Some of the rules in SAOR differ from their versions in *pD**, e.g. $(05_{a,b})$, $(19_{a,b})$. The alert reader may also miss rules to infer transitivity of `rdfs:subClassOf`, `rdfs:subPropertyOf`, `:equivalentClass`, `:equivalentProperty` as well as symmetry of the equivalence and inverse-of properties. Whereas symmetry is covered by symmetric rules $(02_{a,b})$, $(04_{a,b})$, $(05_{a,b})$, transitivity is handled by SAOR via path traversals over internal data structures representing the subclass and subproperty hierarchies, following the RDF ground entailment algorithm outlined in [13].

Note that SAOR does not materialise any axiomatic triples [8]. Axiomatic A-Box statements are not produced by SAOR so as to avoid a bulk of syntactic statements (for example, `rdf:type rdf:Resource` statements and reflexive `:sameAs` statements). Axiomatic T-Box statements are not considered as we have a concretely defined T-Box which is extracted by means of a single scan and does not support updates.

Finally, let us point out that there is good reason for excluding non-standard usage of the ontology vocabulary: non-standard RDF could have unpredictable results even under our simple rule-based entailment. One may consider a finite

[5] $?C_i \in \{?C_1, ..., ?C_n\}$

combination of only four non-standard triples that, upon naive reasoning, would explode all web resources R by inferring $|R|^3$ triples, namely:

```
rdfs:subClassOf rdfs:subPropertyOf rdfs:Resource.
rdfs:subClassOf rdfs:subPropertyOf rdfs:subPropertyOf.
rdf:type rdfs:subPropertyOf rdfs:subClassOf.
rdfs:subClassOf rdf:type :SymmetricProperty.
```

The exhaustive application of standard RDFS inference rules plus standard inference rule for property symmetry together with the typical inference for class membership in `rdfs:Resource` for all collected resources in typical rulesets lead to inference of any possible triple $(r_1\ r_2\ r_3)$ for arbitrary $r_1, r_2, r_3 \in R$.

Having introduced our rule set, we are able to define our notion of closure.

Definition 1 (Closure). *We denote by $Cl_{\mathcal{T}}(\mathcal{A})$ the closure of $\mathcal{A}$, i.e., the union of $\mathcal{A}$ with the set of statements inferred by exhaustive application of rules (01)-(19) from Table 2 with respect to T-Box $\mathcal{T}$.*

2.3 Authoritative Reasoning against Ontology Hijacking

SAOR is designed to counter-act a behaviour we discovered from initial evaluation which we term *ontology hijacking*. We counter such non-authoritative extensions of ontologies by ignoring problematic statements during T-Box generation. Before defining ontology hijacking, let us give some preliminary definitions:

Definition 2 (Authoritative Source).
A graph $s \in \mathcal{KB}$ speaks authoritatively about a concept $c \in \mathcal{C}_{\mathcal{KB}} \cup \mathcal{P}_{\mathcal{KB}}$ if c appears in a triple of s and one of the following holds true:

1. *c is not identified by a URI (i.e., identified by a blank node)*
2. *s is retrievable from a URI which coincides with (or redirects to) the namespace[6] of the URI identifying c.*

Firstly, all sources are authoritative for anonymous classes or properties defined in that source. The second condition is designed to support best practices as currently adopted by web ontology publishers[7].

Let $s \in \mathcal{KB} = (\mathcal{T}, \mathcal{A})$ and $\mathcal{KB}' = (\mathcal{T}', \mathcal{A}') = \mathcal{KB} \setminus \{s\}$ be the knowledge base constructed from all graphs in $\mathcal{KB}$ except s. By *Ontology Hijacking* we now mean that a source s speaks non-authoritatively about a concept $c \in \mathcal{C}_{\mathcal{KB}} \cup \mathcal{P}_{\mathcal{KB}}$ (i.e., where c appears in $\mathcal{T}'$), in such a way that $Cl_{\mathcal{T}}(\mathcal{A}') \neq Cl_{\mathcal{T}'}(\mathcal{A}')$.

Ontology hijacking is the re-definition or extension of a definition of a legacy concept (class or property) in a non-authoritative source such that performing reasoning on legacy A-Box data results in a change in inferencing. One particular method of ontology hijacking is defining new super-concepts of legacy concepts.

[6] Here, slightly abusing XML terminology by "namespace" of a URI we mean the prefix of the URI obtained from stripping off the final NCname

[7] See Appendix A&B of http://www.w3.org/TR/swbp-vocab-pub/

As a concrete example, if one were to publish today a property in an ontology (in a non-authoritative location for FOAF), `my:name`, within which the following was stated: `foaf:name rdfs:subClassOf my:name .`, that person would be hijacking the `foaf:name` property and effecting the translation of all `foaf:name` statements in the web knowledge base into `my:name` statements as well.

Ontology hijacking is problematic in that it vastly increases the amount of statements that are materialised and can potentially harm inferencing on data contributed by other parties. With respect to materialisation, the former issue becomes prominent: instance data published using concepts from popular/core ontologies get translated into a plethora of conceptual models described in obscure ontologies; we quantify the problem in Section 4. However, taking precautions against harmful ontology hijacking is growing more and more important as the Semantic Web features more and more attention; motivation for spamming and other malicious activity propagates amongst certain parties with ontology hijacking being a prospective avenue. With this in mind, we assign sole responsibility for the concepts and thus the semantics of their instances of the concepts to those who maintain the authoritative specification.

Related to the idea of ontology highjacking is the idea of "non-conservative extension" described in the Description Logics literature: cf. [11]. However, the notion of a conservative extension was defined with a slightly different objective in mind: according to the notion of deductively conservative extensions, an ontology O_B is only considered malicious towards O_A if it causes additional inferences with respect to the intersection of the signature of the original O_A with the newly inferred statements. Returning to the `ex:myName` example from above, the superclassing of `foaf:Name` alone would still constitute a conservative extension. However, further stating that `ex:myName a :InversefunctionalProperty.` would indeed violate the conservative extension property, since instances of `ex:myName` might then cause equalities in other remote ontologies as side-effects, independent from the newly defined signature. Summarising, we can state that every non-conservative extension (with respect to our notion of deductive closure) constitutes a case of ontology highjacking, but not vice versa.

In SAOR, we avoid the effects of ontology hijacking and non-conservative extensions by disregarding possibly harmful nonauthoritative use of concepts directly during T-Box construction. Table 1 shows how non-authoritative statements are disregarded upon T-Box construction. The source containing the concept description must be authoritative for the elements highlighted in boldface. Where multiple elements are italicised, at least one such element must be authoritatively spoken for. One can verify from Table 1 and Table 2 that, in the antecedent of each rule, the T-Box axioms must be authoritative for at least one of the class/property names appearing in the A-Box. Thereby, we protect A-Box reasoning from the influence of non-authoritative T-Box axioms. For Rules 02, 04, 05, 09 & 11 if the given source of data is only authoritative for one element, inferencing will only be executed in the direction "away" from that element. For example, for the statements `foaf:Person :equivalentClass ex:NewClass . ex:NewClass :equivalentClass foaf:Person .` described in a source only authoritative for the `ex:` namespace, inferencing will only translate `ex:NewClass` instances into `foaf:Person` and not in the other direction.

When publishing OWL or RDFS descriptions on the Web, we recommend that people avoid ontology-hijacking as defined in this section. We encourage extension of existing concepts where possible so that instance data in the newly published domain get translated into existing domains. Indeed, from brief analysis of some prominent specifications (specifically FOAF, DC, SIOC, SKOS), we found that they were entirely compliant with our restrictive reasoning.

3 Reasoning Algorithm

In the following we firstly present observations on web data that influenced the design of the algorithm, then give an overview of the algorithm, and next discuss details of how we handle T-Box information, perform statement-wise reasoning, and deal with ground equality.

3.1 Characteristics of Web Data

The design of our algorithm is motivated by observations on our Web dataset:

1. Reasoning accesses a large slice of data in the index: around 41% of statements produced uniquely inferred statements.
2. Relative to A-Box data, the volume of T-Box data on the Web is small: only around 2.5% of statements were classifiable as T-Box statements[8].
3. The T-Box is the most frequently accessed segment of data for reasoning: all but Rules 19_{a-d} (`:sameAs`) require access to T-Box information.

Following from the first observation, we employ a file-scan approach which is more efficient in this scenario than query processing lookups. Thus, we avoid the overhead of indexing the data and running full query processing; also we avoid probing the same statements repeatedly for different rules at the low cost of scanning a given percentage of statements not useful for reasoning.

Following from the second and third observations, we optimise by placing T-Box data in a separate data structure accessible by the reasoning engine. Currently, we hold the T-Box data in-memory, but the algorithm can be generalised to provide for an on-disk structure or a distributed in-memory structure as needs require.

3.2 Algorithm Overview

The algorithm involves three scans over the data as illustrated in Figure 1:

1. SCAN 1: separate T-Box information and build in-memory representation
2. PRE SCAN 2: execute rules with only T-Box patterns in the antecedent (Rule 18)
3. SCAN 2: perform reasoning in a statement-wise manner:

[8] Includes RDF collection fragments which may not be part of a class description

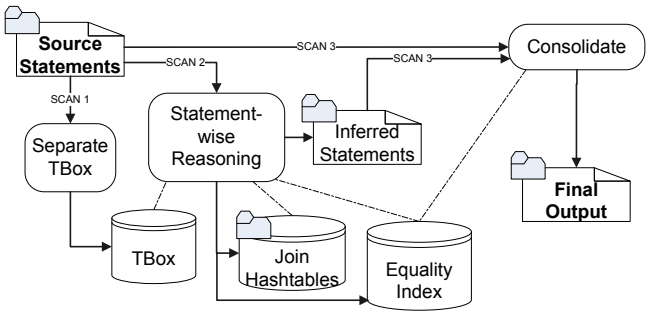

Fig. 1: High-level architecture

- Execute rules with only a single A-Box pattern in the antecedent (Rules 1-11$_a$); join A-Box statement with in-memory T-Box; write inferred statements immediately.
- Execute rules with two or more A-Box patterns in the antecedent (Rules 11$_b$-17); join indices are maintained for such A-Box patterns. When a statement matches one of the A-Box patterns for these rules and the necessary T-Box join exists, the statement is written to the join index and the join index is checked to see if all other A-Box patterns have been previously satisfied; if they have, the rule is fired.
- Execute rules which involve equality reasoning (Rules 19$_{a-d}$); lists of equivalent individuals are maintained in a hashtable structure (equality index). Newly identified equivalences are immediately reflected in the join indices for Rules 11$_b$-17 whereby new A-Box joins may form and fire rules.

4. SCAN 3: consolidate source data along with inferred statements according to the equality index and write to final output

3.3 Handling Structural Data

In the following, we describe how to separate the T-Box data and how to create the data structures for representing the T-Box.

T-Box data from RDFS and OWL specifications can be acquired either from conventional crawling techniques, or by accessing the locations indicated by the dereferenced URIs of classes and properties in the instance data. We assume for brevity that all T-Box data are already present in the input data. If T-Box data are sourced via different means we can build an in-memory representation directly, without requiring the first scan of the input data.

During the scan, all statements relating to the supported T-Box constructs are identified and stored in an in-memory representation of classes and properties. The data structure holds the necessary information to infer new statements given a class or role membership assertion from the A-Box. We employ separate hashtables with URIs as keys and values containing a Java representation of the classes or properties, as follows:

- Property objects contain the property URI and references to objects representing equivalent properties, super properties, inverse properties, domain

classes and range classes. Pointers are also kept to restrictions where the property in question is the object of an `:onProperty` statement.
- Class objects contain the class URI and references to objects representing equivalent classes, super classes and classes for which this class is a component of a union or intersection. On top of these core elements, different types of objects are created for different types of class description:
  - union and intersection classes store references to their constituent class objects
  - enumerated classes store references to constituent individuals
  - restriction classes store a reference to the property the restriction applies to and also, as applicable to the type of restriction:
    * the `:cardinality` or `:maxCardinality` value
    * the class identified by `:allValuesFrom` or `:someValuesFrom`
    * the value of a `:hasValue` restriction

Some class descriptions rely on the `rdf:Collection` construct, namely: unions, intersections and enumerations. To construct in-memory representations of these descriptions, the algorithm performs in-memory joining of `rdf:Collection` segments as the data are scanned according to `rdf:first` and `rdf:rest` properties. Any collections not relevant to the T-Box segment of the knowledge base are discarded at the end of loading the input data.

For each statement, the authority of the source for the given subject and object are inspected. If the statement is allowed as enumerated in Table 1, the statement is added to the T-Box.

3.4 Reasoning by Statement-wise Scan

Having loaded the structural data, the SAOR engine is now prepared for reasoning by statement-wise scan of the data.

We firstly analyse rules which do not require A-Box joins to compute. There are two distinct types of statements which require different handling, namely `rdf:type` statements and general non-`rdf:type` statements. The `rdf:type` statements are subject to class-based entailment reasoning (Rules 1-2 & 9_b-11_a : rules with a single `rdf:type` A-Box pattern), and require joins with class descriptions in the T-Box. The non-`rdf:type` statements are subject to property-based entailments (Rules 3-9_a : rules with a single non-`rdf:type` A-Box pattern) and thus require joins with T-Box property descriptions.

We assume disjointness between the statement categories: we know that the defined semantics of `rdf:type` do not require any property-based entailment and we further do not allow any external extension of the core `rdf:type` semantics (non-standard use). Thus, we do not subject `rdf:type` statements to entailment of Rules 3-9_a.

The reasoning scan process can be described as recursive depth-first reasoning whereby each unique statement produced is input immediately for reasoning. Statements produced thus far for the original input statement are kept in a set to provide uniqueness testing and avoid cycles; a uniquing function is maintained on a resource level ensuring that statements are only produced once for a given

common subject group. Once all of the statements produced by a rule have been themselves recursively analysed, the reasoner moves on to analysing the proceeding rule.

Rules 11_b-17 cannot be computed solely on a statement-wise basis. Instead, for each rule, we assign an on-disk persistent data structure with an in-memory MRU cache. Each index stores a representation of statements which may contribute to satisfying the antecedent of its pertinent rule. During the scan, if a statement satisfies the necessary T-Box join for a rule, it is written to the index for that rule. When a statement is added which completes the pattern of an antecedent for that rule, the rule is fired.

3.5 Equality Reasoning

In the following we discussed `:sameAs` entailment as encoded in Rules 19_{a-d} of Table 2. For Rules $19_{a,b}$, we employ an in-memory index for storing equivalence of individuals. We store the identifiers for equivalent individuals in lists and also store the lists which individuals are in using a hashtable. Thus, we can perform a lookup for an individual in the hashtable to get a reference to a list of equivalent individuals. The list structure maintains the transitivity and symmetric properties of equivalence. Usually, `:sameAs` entailment on individuals results in multiple individuals with the same data attached; however, we select a "pivot element" to reduce the number of inferred statements. The pivot element of each list is used to keep a consistent identifier for the set of equivalent individuals: the first one encountered is chosen. For alternative choices of pivot identifiers on web data see [9]. We use the pivot identifier to consolidate data by rewriting all occurrences of equivalent identifiers to the pivot identifier (effectively merging the equivalent set into one individual).

The in-memory equivalence index is filled from raw input `:sameAs` statements and also from inferencing performed on Rules 12, 13 and 17. For the purposes of the A-Box scan, we need not be immediately concerned about equality reasoning for closure: no joins are present on the individual level. However, for the join index reasoning, equality reasoning is paramount for closure. The join indices are immediately updated to reflect new equality knowledge; identifiers for equivalent individuals are rewritten to their pivot identifiers are soon as equivalence is determined. Rewriting of indices can lead to new inferences whereby the rewritten identifiers align under the pivot identifier to form a new join, thus firing a rule.

Based on the equivalence knowledge attained during the second scan, the inferred output and input data are finally scanned once more to ensure proper consolidation. All statements are rewritten so that they only contain pivot identifiers, producing the final output.

4 Evaluation and Discussion

We now provide evaluation of the SAOR methodology, firstly with quantitative analysis of the importance of authoritative reasoning, and secondly we provide some performance measurements, discussion and some insights into the fecundity

of each rule wrt. reasoning over web data. Throughout, we use a 106M statement web-crawl dataset from mid-April 2008, taken from 315k sources.

To show the effects of ontology hijacking we constructed two T-Boxes with and without authoritative analysis. We then ran reasoning on single membership assertions for the top five classes and properties found natively in our dataset. Table 3 summarises the results. Taking `foaf:Person` as an example, with an authoritative T-Box, six statements are output for every input `rdf:type foaf:Person` statement. With the non-authoritative T-Box, 362 statements are output for every such input statement. Considering that there are 2.4M such statements in the input dataset, overall output for `rdf:type foaf:Person` input statements alone approach 1 billion statements for non-authoritative reasoning. With authoritative reasoning, we only produce 14M output statements: a 64.8x savings on materialised statements.

Class URI	$\mathcal{A}$	$\mathcal{NA}$	n	$n * \mathcal{A}$	$n * \mathcal{NA}$
http://purl.org/rss/1.0/item	0	356	2,550,664	0	908,036,384
http://xmlns.com/foaf/0.1/Person	6	389	2,410,331	14,461,986	937,618,759
http://xmlns.com/foaf/0.1/Document	1	355	1,497,132	1,497,132	531,481,860
http://xmlns.com/wordnet/1.6/Person	0	236	1,097,415	0	258,989,940
http://xmlns.com/foaf/0.1/chatEvent	0	0	1,097,265	0	0
TOTAL	7	1,336	8,652,807	15,959,118	2,636,126,943
Property URI	$\mathcal{A}$	$\mathcal{NA}$	n	$n * \mathcal{A}$	$n * \mathcal{NA}$
http://purl.org/dc/elements/1.1/title	0	250	4,222,957	0	1,055,739,250
http://xmlns.com/foaf/0.1/name	5	664	3,753,791	18,768,955	2,492,517,224
http://purl.org/dc/elements/1.1/date	0	625	3,677,251	0	2,298,281,875
http://xmlns.com/foaf/0.1/nick	0	637	3,100,733	0	1,975,166,921
http://purl.org/dc/elements/1.1/description	0	631	30,138,087	0	19,017,132,897
TOTAL	5	2,807	44,892,819	18,768,955	26,838,838,167

Table 3: Comparison of authoritative and non-authoritative reasoning for the number of inferred statements produced w.r.t. the five most frequently occurring classes and properties in the input data.

We measured the performance of applying only the rules which do not require A-Box joins (1-11_a) and for applying all rules. The results of the evaluation on a 2.2 GhZ AMD Opteron machine with 3G of Java heap-space is shown in Figure 2. Please note that the trend with respect to statements read/input statements processed is very similar to that presented for written statements/statements output. Also, please observe that applying rules without A-Box joins exhibits perfectly linear scaling behaviour, while using all rules slows down the algorithm after inferring about 120m output statements. For implementing the on-disk A-Box join indices we employ BerkeleyDB[9], which slows down considerably if the index size exceeds a certain limit (depending on caching policy and main memory available to the JVM). We are currently investigating alternatives to dynamic data structures for Rules 11_b-17.

Table 4 lists the number of times the rules for a given primitive were fired during reasoning over all rules. Interestingly, from Figure 2 and Table 4 we can deduce that the bulk of current web reasoning is covered by those rules (1-12_a) which exhibit linear scale.

[9] http://www.oracle.com/database/berkeley-db/je/

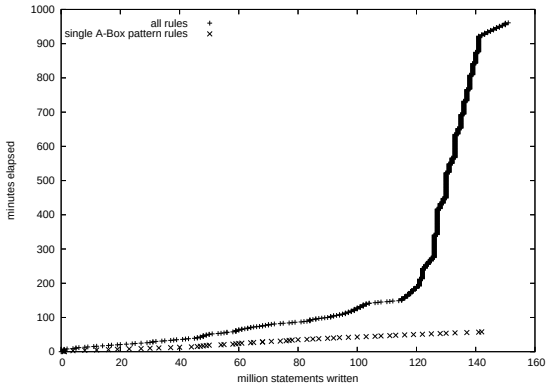

Fig. 2: Performance of the inferencing algorithm.

No	Primitive	Inferred Count
01	rdfs:subClassOf	66,283,568
02	:equivalentClass	7,325,048
03	rdfs:subPropertyOf	7,314,815
04	:equivalentProperty	6,943,803
05	:inverseOf	8,485,414
06	rdfs:domain	29,850,430
07	rdfs:range	19,466,468
08	:SymmetricProperty	458,467
09	:hasValue	9,938
10	:unionOf	5,676,861
11	:intersectionOf	13,239
12	:FunctionalProperty	15,218
13	:InverseFunctionalProperty	1,379,003
14	:TransitiveProperty	2,862,631
15	:someValuesFrom	51,403
16	:allValuesFrom	460,031
17	:cardinality	265
18	:oneOf	5,898

Table 4: Count of number of statements inferred for each primitive.

5 Related Work

OWL reasoning, specifically Query Answering over OWL Full, is not tackled by typical DL Reasoners; such as FaCT++ [18], RACER [7] or Pellet [15]; which focus on complex reasoning tasks such as subsumption checking and provable completeness of reasoning. Likewise, KAON2 [12], which reports better results on query answering, is limited to OWL-DL expressivity due to completeness requirements. Despite being able to deal with complex ontologies in a complete manner, these systems are not tailored for the particular challenges of processing large amounts of RDF data.

Conversely, incomplete (wrt. OWL Full) rule-based inference, as we advocate it in this paper, may be considered to have greater potential for scale. Several rule expressible non-standard OWL fragments; namely OWL-DLP [6], OWL$^-$ [5] (which is a slight extension of OWL DLP), OWLPrime [20], *pD** [17], or *intentional OWL* [4, Section 9.3]; have been defined in the literature and enable incomplete but sound RDFS and OWL inferences. Amongst those fragments, the fragment we support here is most closest to [17].

Systems such as Triple [14], JESS[10], or Jena[11] support rule representable RDFS or OWL fragments as we do, but only work in-memory whereas our framework is focused on conducting scalable reasoning using persistent storage.

Analogous to our approach, [3] introduce certain restrictions for axioms accepted by a reasoning engine, however, lacking a rigorous treatment of acceptable axioms.

The OWLIM [10] family of systems allows reasoning over a number of rule-representable OWL fragments using the TRREE: Triple Reasoning and Rule Entailment Engine. Besides the in-memory version SwiftOWLIM, which uses TRREE, there is also a version offering query-processing over a persistent image of the repository, BigOWLIM, which comes closest technically to our approach despite focusing on different fragments of OWL, including those inferring inconsistencies. Whereas similarly to BigOWLIM, we employ persistent materialisation of inferred triples, our reasoning approach strictly focuses on sensible reasoning for web data; we only consider a positive fragment of OWL-Horst and analyse the authority of T-Box statements. We deliberately sacrifice logical completeness for what we believe to be a more cautious, but still sound approach for the web data use-case.

6 Conclusion and Future Work

We have presented SAOR: a reasoning methodology for performing reasoning over Web data based on primitives known to scale. To keep the resulting knowledge base manageable, both in size and quality, we made the following modifications to traditional reasoning procedures:

- allow only standard use of RDF and disallow metamodelling
- allow extension of classes and properties only from authoritative sources (no ontology hijacking)
- use pivot identifiers instead of full materialisation of equality

We envision extensions to our system along two lines: scalability enhancements by replacing the dynamic on-disk data structure with a more scalable scans/sort approach and distributing the system using a hash-based placement or T-Box replication strategy.

References

1. D. Brickley and R. Guha. Rdf vocabulary description language 1.0: Rdf schema. W3C Recommendation, Feb. 2004. http://www.w3.org/TR/rdf-schema/.
2. J. d. Bruijn and S. Heymans. Logical foundations of (e)RDF(S): Complexity and reasoning. In *6th International Semantic Web Conference*, number 4825 in LNCS, pages 86–99, Busan, Korea, Nov 2007.
3. G. Cheng, W. Ge, H. Wu, and Y. Qu. Searching semantic web objects based on class hierarchies. In *Proceedings of Linked Data on the Web Workshop*, 2008.

[10] http://herzberg.ca.sandia.gov/

[11] http://jena.sourceforge.net/

4. J. de Bruijn. *Semantic Web Language Layering with Ontologies, Rules, and Meta-Modeling*. PhD thesis, University of Innsbruck, 2008.
5. J. de Bruijn, A. Polleres, R. Lara, and D. Fensel. OWL$^-$. Final draft d20.1v0.2, WSML, 2005.
6. B. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *13th International Conference on World Wide Web*, 2004.
7. V. Haarslev and R. Möller. Racer: A core inference engine for the semantic web. In *International Workshop on Evaluation of Ontology-based Tools*, 2003.
8. P. Hayes. RDF Semantics. W3C Recommendation, Feb. 2004. http://www.w3.org/TR/rdf-mt/.
9. A. Hogan, A. Harth, and S. Decker. Performing object consolidation on the semantic web data graph. In *1st I3 Workshop: Identity, Identifiers, Identification Workshop*, 2007.
10. A. Kiryakov, D. Ognyanov, and D. Manov. Owlim - a pragmatic semantic repository for owl. In *Web Information Systems Engineering Workshops*, LNCS, pages 182–192, New York, USA, Nov 2005.
11. C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *Proc. of IJCAI-2007*, pages 453–459, 2007.
12. B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Forschungszentrum Informatik, Karlsruhe, Germany, 2006.
13. S. Muñoz, J. Pérez, and C. Gutiérrez. Minimal deductive systems for rdf. In *ESWC*, pages 53–67, 2007.
14. M. Sintek and S. Decker. Triple - a query, inference, and transformation language for the semantic web. In *1st International Semantic Web Conference*, pages 364–378, 2002.
15. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.
16. M. K. Smith, C. Welty, and D. L. McGuinness. OWL Web Ontology Language Guide. W3C Recommendation, Feb. 2004. http://www.w3.org/TR/owl-guide/.
17. H. J. ter Horst. Combining rdf and part of owl with rules: Semantics, decidability, complexity. In *4th International Semantic Web Conference*, pages 668–684, 2005.
18. D. Tsarkov and I. Horrocks. Fact++ description logic reasoner: System description. In *International Joint Conf. on Automated Reasoning*, pages 292–297, 2006.
19. T. D. Wang, B. Parsia, and J. A. Hendler. A survey of the web ontology landscape. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, pages 682–694, Athens, GA, USA, Nov. 2006.
20. Z. Wu, G. Eadon, S. Das, E. I. Chong, V. Kolovski, M. Annamalai, and J. Srinivasan. Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle. In *24th International Conference on Data Engineering*. IEEE, 2008. To appear.