Scalable OWL 2 Reasoning for Linked Data*

Aidan Hogan[1], Jeff Z. Pan[2], Axel Polleres[1], and Yuan Ren[2]

[1] Digital Enterprise Research Institute, National University of Ireland, Galway
aidan.hogan@deri.org
axel.polleres@deri.org
[2] Department of Computing Science, University of Aberdeen
jeff.z.pan@abdn.ac.uk
y.ren@abdn.ac.uk

Abstract. The goal of the Scalable OWL 2 Reasoning for Linked Data tutorial is twofold: first, to introduce scalable reasoning and querying techniques to SW researchers as powerful tool to make use of linked data and large-scale ontologies, and second, to present interesting research problems for SW that arise in dealing with TBox and ABox reasoning in OWL 2. The tutorial consists of three parts. It will begin with an introduction of linked data, as well as its relationship with ontologies, such as the ones in OWL2. It will then start with the instance retrieval reasoning service, introducing how to provide scalable reasoning services for OWL 2 RL (a tractable fragment of OWL 2). The third part of the tutorial will present recent work on faithful approximate reasoning for OWL2-DL

1 Introduction

Over the past few years, various Web publishers have turned to RDF and Linked Data principles as a means of disseminating information in a machine-interpretable way, resulting in a burgeoning Web of Data which now includes interlinked content provided by corporate bodies (e.g., BBC [48], BestBuy [31], New York Times[3], Freebase[4]), community-driven efforts (e.g., WIKIPEDIA/DBpedia[5] [13]), social networking sites (e.g., hi5[6], LiveJournal[7]), biomedical datasets (e.g., DrugBank[8], Linked Clinical Trials[9]), governmental entities (e.g., `data.gov.uk`, `data.gov`), academia (e.g., DBLP[10], UniProt[11]), as well as some esoteric corpora (e.g., Poképédia[12], Linked Open Num-

* The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2), by an IRCSET Scholarship, by the EU MOST project, and by the EPSRC LITRO project.

[3] `http://data.nytimes.com/`; retr. 15/10/2010
[4] `http://www.freebase.com/`; retr. 15/10/2010
[5] `http://dbpedia.org/`; retr. 15/10/2010
[6] `http://api.hi5.com/`; retr. 15/10/2010
[7] `http://livejournal.com`; retr. 15/10/2010
[8] `http://www.drugbank.ca/`; retr. 15/10/2010
[9] `http://linkedct.org/`; retr. 15/10/2010
[10] `http://www4.wiwiss.fu-berlin.de/dblp/`; retr. 15/10/2010
[11] `http://www.uniprot.org/`; retr. 15/10/2010
[12] `http://www.pokepedia.net/`; retr. 15/10/2010

bers[13] [84]). See http://lod-cloud.net (retr. 15/10/2010) for Cyganiak and Jentzsch's Linked Open Data cloud diagram which illustrates the datasets comprising the current (and past) Web of Data. (*Please see [2] in these proceedings for extensive introduction and discussion relating to Linked Data and the Web of Data.*)

As such, there now exists a rich vein of heterogeneous, structured and interlinked data on the Web. To enable interoperability and subsequent data integration, Linked Data literature encourages reuse of URIs—particularly those referential to classes and properties (schema-level *terminology*)—across data sources: in the ideal case, a Linked Data consumer can perform a simple (RDF-)merge of datasets, where consistent use of terminology ensures that resources are described uniformly and thus can be accessed and queried uniformly. Although this ideal is achievable in part by community agreement and self-organising phenomena such as preferential attachment [6]—whereby, for example, the most popular classes and properties would become the de-facto consensus and thus more widely used—given the ad-hoc decentralised nature of the Web, complete and appropriate agreement upon the broad spectrum of terminology needed to fully realise the Web of Data is probably infeasible.

Instead, Linked Data publishers may use different but analogous terminology to describe their data: competing vocabularies may offer different levels of granularity or expressivity more suitable to a given publisher's needs, may be popular at different times or within different communities, etc. Publishers may not only choose different vocabularies, but may also choose alternate terms within a given vocabulary to model analogous information; for example, vocabularies may offer pairs of *inverse properties*—e.g., foaf:made/foaf:maker—which poses the publisher with two options for stating the same information (and where stating both could be considered redundant). Further still, publishers may "cherry-pick" vocabularies, choosing a heterogeneous "bag of terms" to describe their data [12].

This becomes a significant obstacle for applications consuming a sufficiently heterogeneous corpus: for example, queries posed against the data must emulate the various terminological permutations possible to achieve (more) complete answers. Here, we take the motivating example of a simple query described in prose as:

What are the webpages related to ex:resource***?***

Knowing that the property foaf:page is commonly used in Linked Data to define the relationship from resources to the documents somehow concerning them, we can formulate a simple structured query in SPARQL [66]—the W3C standardised RDF query language—as given in Listing 1. (*Please see [22] in these proceedings for an introduction to SPARQL, and further discussion on combining RDFS and OWL entailment with SPARQL.*)

However, within Linked Data, there exist various other, more fine-grained properties for relating a resource to specific types of pages—these properties are not only given by FOAF, but also by remote vocabularies. Thus, to ensure more complete answers, the SPARQL query must use disjunction (UNION clauses) to reflect the possible triples which may answer the query; we give such an example in Listing 2 involving properties we found in a corpus of one billion triples of Linked Data crawled in May 2010, where

[13] http://km.aifb.kit.edu/projects/numbers/; retr. 15/10/2010

Listing 1. Simple query for all pages relating to `ex:resource`

```
SELECT ?page
WHERE {
  ex:resource foaf:page ?page .
}
```

we additionally annotate each pattern with the total number of triples found in our corpus for the respective predicate; this gives a *rough* indicator of the relative likelihood of finding additional answers with each additional pattern.

Not only is the resulting query much more cumbersome to formulate, but it also requires a much more in-depth knowledge of the various vocabularies in the corpus. However, since `foaf:page` is relatively well-known within the Linked Data community, all of the properties appearing in the extended query are (possibly indirectly) related to `foaf:page` using RDFS and OWL connectives in their respective vocabularies. In fact, all properties referenced in Listing 2 are chosen on the basis that they are directly or indirectly related to `foaf:page` by `rdfs:subPropertyOf` or `owl:inverseOf`, where relations using these properties can be used to infer `foaf:page` answers —note that we italicise patterns for properties which have an *inverse* (sub-)relation to `foaf:page` in Listing 2. Thus, given these ad-hoc mappings provided by the Linked Data publishers themselves, we have the necessary formal knowledge to be able to answer the former simple query with all of the answers given by the latter elaborate query. In order to exploit this knowledge and realise this goal in the general case, we require some form of *reasoning*. (Of course, this problem of *heterogeneity* is not purely specific to SPARQL querying, but likewise extends to many other consumer applications or techniques operating over a corpus of Linked Data.)

Along these, in this tutorial, we look at two different complementary reasoning approaches:

- in §2, we look at SAOR: a lightweight, pragmatic rule-based materialisation engine which applies a scalable subset of OWL 2 RL/RDF, is designed to operate over a cluster of commodity hardware in a distributed setting with little or no coordination between machines, and conservatively discards terminological knowledge given by unverifiable sources;
- in § 3, we describe a system for performing approximative, but relatively expressive TBox reasoning with respect to OWL 2 DL, where $\mathcal{SROIQ}$ ontologies are simplified into $\mathcal{EL}^{++}$, and where the approximate representation is then classified using scalable techniques which typically demonstrate high recall when compared with the non-approximative classification.

In particular, we note that the rule-based reasoning described in §2 offers linear scalable with respect to assertional (instance) data, but is rather inexpressive, especially when dealing with terminological knowledge. Conversely, the approximative reasoning of § 3 offers expressive PTIME-complete reasoning over terminological knowledge, but this is still a memory based approach.

Listing 2. Extended query for all pages relating to `ex:resource`

```
SELECT ?page
WHERE {
  { ex:resource foaf:page ?page . }                    #4,923,026
  UNION { ex:resource foaf:weblog ?page . }           #10,061,003
  UNION { ex:resource foaf:homepage ?page . }          #9,522,912
  UNION {?page foaf:topic ex:resource . }              #6,163,769
  UNION {?page foaf:primaryTopic ex:resource . }       #3,689,888
  UNION { ex:resource mo:musicbrainz ?page . }           #399,543
  UNION { ex:resource foaf:openid ?page . }              #100,654
  UNION { ex:resource foaf:isPrimaryTopicOf ?page . }     #92,927
  UNION { ex:resource mo:wikipedia ?page . }              #55,228
  UNION { ex:resource mo:myspace ?page . }                #28,197
  UNION { ex:resource po:microsite ?page . }              #15,577
  UNION { ex:resource mo:amazon_asin ?page . }            #14,790
  UNION { ex:resource mo:imdb ?page . }                    #9,886
  UNION { ex:resource mo:fanpage ?page . }                 #5,135
  UNION { ex:resource mo:biography ?page . }               #4,609
  UNION { ex:resource mo:discogs ?page . }                 #1,897
  UNION { ex:resource rail:arrivals ?page . }                #347
  UNION { ex:resource rail:departures ?page . }              #347
  UNION { ex:resource mo:musicmoz ?page . }                  #227
  UNION { ex:resource mo:discography ?page . }               #195
  UNION { ex:resource mo:review ?page . }                     #46
  UNION { ex:resource mo:freedownload ?page . }               #37
  UNION { ex:resource mo:mailorder ?page . }                  #35
  UNION { ex:resource mo:licence ?page . }                    #28
  UNION { ex:resource mo:paiddownload ?page . }               #13
  UNION { ex:resource foaf:tipjar ?page . }                    #8
  UNION { ex:resource doap:homepage ?page . }                  #1
  UNION { ex:resource doap:old-homepage ?page . }              #1
  UNION { ex:resource mo:download ?page . }                    #0
  UNION { ex:resource mo:event_homepage ?page . }              #0
  UNION { ex:resource mo:free_download ?page . }               #0
  UNION { ex:resource mo:homepage ?page . }                    #0
  UNION { ex:resource mo:paid_download ?page . }               #0
  UNION { ex:resource mo:preview_download ?page . }            #0
  UNION { ex:resource mo:olga ?page . }                        #0
  UNION { ex:resource mo:onlinecommunity ?page . }             #0
  UNION { ex:resource plink:addFriend ?page . }                #0
  UNION { ex:resource plink:atom ?page . }                     #0
  UNION { ex:resource plink:content ?page . }                  #0
  UNION { ex:resource plink:foaf ?page . }                     #0
  UNION { ex:resource plink:profile ?page . }                  #0
  UNION { ex:resource plink:rss ?page . }                      #0
  UNION { ex:resource xfn:mePage ?page . }                     #0
}
```

2 Scalable, Incomplete, OWL 2 RL/RDF Rule-Based Reasoning

Herein, we detail our system for performing reasoning over large-scale Linked Data corpora, which we call the *Scalable Authoritative OWL Reasoner* (SAOR) [35, 37, 34]. Our use-case scenario is the Semantic Web Search Engine (SWSE) [36],[14] which offers search and browsing over datasets consisting of approximately one billion Linked Data triples crawled from the Web: we want to use reasoning to materialise inferences and make them available in the SWSE results presented to users.

To validate our approach, we take a corpus of 1.12 billion quadruples (947 million unique triples) crawled from 4 million RDF/XML documents in May 2010; the data were taken from 783 different pay-level-domains (PLDS—e.g., `dbpedia.org`). Our crawl was open (not domain restricted) and all HTTP URIs appearing in the data (except those with filtered non-RDF/XML file extensions) were considered as candidates for crawling. More details on this corpus are available in [34].

Thus, in this section we look at applying incomplete OWL 2 RL/RDF materialisation in a manner sympathetic with our use-case, over static corpora of unverified Linked Data collected from millions of sources, consisting of approximately one billion input facts. In particular:

- we begin by discussing the requirements of our system for performing Linked Data/Web reasoning and high-level design decisions (§ 2.2);
- we continue by discussing a separation of terminological data during reasoning, providing soundness and conditional completeness results (§ 2.3);
- we subsequently detail the generic optimisations that a separation of terminological data allows (§ 2.4);
- we then look at identifying a subset of OWL 2 RL/RDF rules suitable for scalable materialisation, and discuss our method for performing distributed *authoritative* reasoning over our Linked Data corpus (§ 2.5);
- finally, we provide an overview of related work (§ 2.7) and give general discussion (§ 2.8).

2.1 Preliminaries

Before we continue, we briefly give some necessary preliminaries relating to (i) RDF, (ii) Linked Data principles, and (iii) rules.

RDF We briefly give some necessary notation relating to RDF constants and RDF triples; see [30].

RDF Constant Given the set of URI references U, the set of blank nodes B, and the set of literals L, the set of *RDF constants* is denoted by $\mathsf{C} := \mathsf{U} \cup \mathsf{B} \cup \mathsf{L}$. We interpret blank-nodes as skolem constants signifying particular individuals, as opposed to existential variables as prescribed by the RDF Semantics [30]; also, we rewrite blank-node labels when merging documents to ensure uniqueness of labels across those documents [30]. Finally, note that we may use '`a`' as a shortcut for `rdf:type`, following convention in Turtle [7].

[14] Prototype available at `http://swse.deri.org/`

RDF Triple A triple $t := (s, p, o) \in (\mathsf{U} \cup \mathsf{B}) \times \mathsf{U} \times \mathsf{C}$ is called an *RDF triple*, where *s* is called subject, *p* predicate, and *o* object. A triple $t := (s, p, o) \in \mathsf{G}, \mathsf{G} := \mathsf{C} \times \mathsf{C} \times \mathsf{C}$ is called a *generalised triple* [24], which allows any RDF constant in any triple position: henceforth, we assume generalised triples unless explicitly stated otherwise. We call a finite set of triples $G \subset \mathsf{G}$ a *graph*.

RDF Variable/RDF Triple Pattern We denote the set of all *RDF variables* as V; we call a generic member of the set $\mathsf{V} \cup \mathsf{C}$ an *RDF term*. Again, we denote RDF variables as alphanumeric strings with a '`?`' prefix. We call a triple of RDF terms—where variables are allowed in any position—an *RDF triple pattern*.

Variable Substitution We call a mapping from the set of variables to the set of constants $\theta : \mathsf{V} \to \mathsf{C}$ a *variable substitution*; we denote the set of all such substitutions by Θ.

Linked Data Principles and Provenance In order to cope with the unique challenges of handling diverse and unverified Web data, many of our components and algorithms require inclusion of a notion of provenance: consideration of the source of RDF data found on the Web. Thus, herein we provide some formal preliminaries for the Linked Data principles, and HTTP mechanisms for retrieving RDF data. (*Please see [2] in these proceedings for extensive introduction and discussion relating to Linked Data and the Web of Data.*)

Linked Data Principles Herein, we will refer to the four best practices of Linked Data as follows [9]:

- (**LDP1**) use URIs as names for things;
- (**LDP2**) use HTTP URIs so those names can be dereferenced;
- (**LDP3**) return useful information upon dereferencing of those URIs; and
- (**LDP4**) include links using externally dereferenceable URIs.

Data Source We define the *http-download* function $\mathsf{get} : \mathsf{U} \to 2^{\mathsf{G}}$ as the mapping from a URI to an RDF graph it provides by means of a given HTTP lookup [20] which directly returns status code `200 OK` and data in a suitable RDF format, or to the empty set in the case of failure; this function also performs a rewriting of blank-node labels (based on the input URI) to ensure uniqueness when merging RDF graphs [30]. We define the set of *data sources* $\mathsf{S} \subset \mathsf{U}$ as the set of URIs $\mathsf{S} := \{s \in \mathsf{U} \mid \mathsf{get}(s) \neq \emptyset\}$.

RDF Triple in Context/RDF Quadruple An ordered pair (t, c) with a triple $t := (s, p, o)$, and with a context $c \in \mathsf{S}$ and $t \in \mathsf{get}(c)$ is called a *triple in context c*. We may also refer to (s, p, o, c) as an *RDF quadruple* or quad q with context c.

HTTP Redirects/Dereferencing A URI may provide a HTTP redirect to another URI using a `30x` response code [20]; we denote this function as $\mathsf{redir} : \mathsf{U} \to \mathsf{U}$ which may map a URI to itself in the case of failure (e.g., where no redirect exists). We denote the fixpoint of redir as redirs, denoting traversal of a number of redirects (a limit may be set on this traversal to avoid cycles and artificially long redirect paths). We define

dereferencing as the function $\mathsf{deref} := \mathsf{get} \circ \mathsf{redirs}$ which maps a URI to an RDF graph retrieved with status code `200 OK` *after* following redirects, or which maps a URI to the empty set in the case of failure.

Atoms and Rules In this section, we briefly introduce some notation as familiar particularly from the field of Logic Programming [52], which eventually gives us our notion of a *rule*. As such, much of the notation in this section serves as a generalisation of the RDF notation already presented; we will discuss this relation as pertinent.

Atom An *atomic formula* or *atom* is a formula of the form $p(e_1, \ldots, e_n)$, where all such $e_1, \ldots, e_n$ are terms (like Datalog, function symbols are disallowed) and where p is a *predicate* of arity n—we denote the set of all such atoms by Atoms. As such, this notation can be thought of as generalising that of RDF triples, where we use a standard RDF *ternary predicate* T to represent RDF triples in the form $T(s, p, o)$—for example, T`(Fred, age, 56)`—where we will typically leave T implicit.

Note that a term e_i can also be a variable, and thus RDF triple patterns can also be represented directly as atoms. Atoms not containing variables are called *ground atoms* or simply *facts*, denoted as the set Facts (a generalisation of G); a finite set of facts I is called a (Herbrand) *interpretation* (a generalisation of a graph). Letting A and B be two atoms, we say that A *subsumes* B—denoted $A \triangleright B$—if there exists a substitution $\theta \in \Theta$ of variables such that $A\theta = B$ (applying θ to the variables of A yields B); we may also say that B is an *instance* of A; if B is ground, we say that it is a *ground instance*. Similarly, if we have a substitution $\theta \in \Theta$ such that $A\theta = B\theta$, we say that θ is a *unifier* of A and B; we denote by $\mathsf{mgu}(A, B)$ the *most general unifier* of A and B which provides the "minimal" variable substitution (up to variable renaming) required to unify A and B.

Rule A *rule* R is given as follows:

$$H \leftarrow B_1, \ldots, B_n (n \geq 0)\,, \tag{1}$$

where $H, B_1, \ldots, B_n$ are atoms, H is called the *head* (conclusion/consequent) and $B_1, \ldots, B_n$ the *body* (premise/antecedent). We use $\mathsf{Head}(R)$ to denote the head H of R and $\mathsf{Body}(R)$ to denote the body $B_1, \ldots, B_n$ of R.[15] The variables of our rules are *range restricted*, also known as *safe* [80]: like Datalog, the variables appearing in the head of each rule must also appear in the body, which means that a substitution which grounds the body must also ground the head. We denote the set of all such rules by Rules. A rule with an empty body is considered a fact; a rule with a non-empty body is called a *proper-rule*. We call a finite set of such rules a *program* P.

Like before, a ground rule is one without variables. We denote with $\mathsf{Ground}(R)$ the set of ground instantiations of a rule R and with $\mathsf{Ground}(P)$ the ground instantiations of all rules occurring in a program P. Again, an ***RDF rule*** is a specialisation of the above rule, where atoms strictly have the ternary predicate T and contain RDF terms; an ***RDF program*** is one containing RDF rules, etc.

[15] Such a rule can be represented as a definite Horn clause.

Note that we may find it convenient to represent rules as having multiple atoms in the head, such as:

$$H_1, \ldots, H_m (m \geq 1) \leftarrow B_1, \ldots, B_n (n \geq 0) \,,$$

where we imply a *conjunction* between the head atoms, such that this can be equivalently represented as the set of rules:

$$\{H_i \leftarrow B_1, \ldots, B_n \mid (1 \leq i \leq m)\} \,.$$

Immediate Consequence Operator We give the immediate consequence operator $\mathfrak{T}_P$ of a program P under interpretation I as:[16]

$$\mathfrak{T}_P : \ 2^{\mathsf{Facts}} \rightarrow 2^{\mathsf{Facts}}$$
$$I \mapsto \left\{ \mathsf{Head}(R)\theta \mid R \in P \wedge \exists I' \subseteq I \text{ s.t. } \theta = \mathsf{mgu}\big(\mathsf{Body}(R), I'\big) \right\}$$

Intuitively, the immediate consequence operator maps from a set of facts I to the set of facts it directly entails with respect to the program P—note that $\mathfrak{T}_P(I)$ will retain the facts in P since facts are rules with empty bodies and thus unify with any interpretation, and note that $\mathfrak{T}_P$ is *monotonic*—the addition of facts and rules to a program can only lead to the same or additional consequences. We may refer to the application of a single rule $\mathfrak{T}_{\{R\}}$ as a *rule application*.

Since our rules are a syntactic subset of Datalog, $\mathfrak{T}_P$ has a *least fixpoint*—denoted $\mathsf{lfp}(\mathfrak{T}_P)$—which can be calculated in a bottom-up fashion, starting from the empty interpretation Δ and applying iteratively $\mathfrak{T}_P$ [87] (here, convention assumes that P contains the set of input facts as well as proper rules). Define the iterations of $\mathfrak{T}_P$ as follows: $\mathfrak{T}_P \uparrow 0 = \Delta$; for all ordinals α, $\mathfrak{T}_P \uparrow (\alpha + 1) = \mathfrak{T}_P(\mathfrak{T}_P \uparrow \alpha)$; since our rules are Datalog, there exists an α such that $\mathsf{lfp}(\mathfrak{T}_P) = \mathfrak{T}_P \uparrow \alpha$ for $\alpha < \omega$, where ω denotes the least infinite ordinal—i.e., the immediate consequence operator will reach a fixpoint in countable steps [80]. Thus, $\mathfrak{T}_P$ is also *continuous*. We call $\mathsf{lfp}(\mathfrak{T}_P)$ *the least model*, or the *closure* of P, which is given the more succinct notation $\mathsf{lm}(P)$.

2.2 Linked Data Reasoning: Overview

Performing reasoning over large amounts of arbitrary RDF data sourced from the Web implies unique challenges which have not been significantly addressed by the literature. Given that we will be dealing with a corpus in the order of a billion triples collected from millions of unvetted sources, we must acknowledge two primary challenges:

- **scalability**: the reasoning approach must scale to billion(s) of statements;
- **robustness**: the reasoning approach should be tolerant to noisy, impudent and inconsistent data.

These requirements heavily influence the design choices of our reasoning approach, where in particular we (must) opt for performing reasoning which is incomplete with respect to OWL semantics.

[16] Note that in our Herbrand semantics, an interpretation I can be thought of as simply a set of facts.

Incomplete Reasoning: Rationale Current standard RDFS/OWL reasoning approaches are not naturally suited to meet the aforementioned challenges.

Firstly, standard RDFS entails infinite triples, although implementations commonly support a decidable (finite) subset [76, 61, 62, 86]. In any case, RDFS does not support reasoning over OWL axioms commonly provided by Linked Data vocabularies.

With respect to OWL, reasoning with respect to OWL (2) Full is known to be undecidable. Reasoning with standard dialects such as OWL (2) DL or OWL Lite have more than exponential worst-case complexity, and are typically implemented using tableau-based algorithms which have yet to demonstrate scalability for reasoning over assertional data which would be propitious to our scenario: certain reasoning tasks may require satisfiability checking which touch upon a large proportion of the individuals in the knowledgebase, and may have to operate over a large, branching search space [4]. Similarly, although certain optimisation techniques may make the performance of such tableau-reasoning sufficient for certain *reasonable* inputs and use-cases, guarantees of such *reasonability* do not extend to a Web corpus like ours. Reasoning with respect to the new OWL 2 profiles—viz., OWL 2 EL/QL/RL—have polynomial runtime, which although an improvement, may still be prohibitively expensive for our scenario.

Aside from complexity considerations, most OWL documents on the Web are in any case OWL Full: "syntactic" assumptions made in DL-based profiles are violated by even very commonly used ontologies. For example, the FOAF vocabulary knowingly falls into OWL Full since, e.g., `foaf:name` is defined as a sub-property of the core RDFS property `rdfs:label`, and `foaf:mbox_sha1sum` is defined as a member of both `owl:InverseFunctionalProperty` and `owl:DatatypeProperty`: such axioms are disallowed by OWL (2) DL (and by extension, disallowed by the sub-dialects and profiles).

Finally, OWL semantics prescribe that anything can be entailed from an inconsistency, following the *principle of explosion* in classical logics. This is not only true of OWL (2) Full semantics, but also of those sub-languages rooted in Description Logics, where reasoners check entailment by reduction to satisfiability—if the original graph is inconsistent, it is already in itself unsatisfiable, and the entailment check will return true for any arbitrary graph. Given that consistency cannot be expected on the Web, we wish to avoid the arbitrary entailment of all possible triples from our knowledge-base. Along these lines, a number of paraconsistent reasoning approaches have been defined in the literature (see, e.g., [41, 55, 89, 56, 51]) typically relying upon four-valued logic [8]—however, again, these approaches have yet to demonstrate the sort of performance required for our scenario.

Thus, due to the inevitability of inconsistency and the prohibitive computational complexity involved, complete reasoning with respect to the standardised RDFS/OWL (sub-)languages is infeasible for our scenario. We instead argue that completeness (with respect to the language) is not a requirement for our use-case, particularly given that the corpus itself represents incomplete knowledge (cf. [19, 32]).

Moving forward, we opt for sound but incomplete support of OWL Full semantics such that entailment is axiomatised by a set of rules which are applicable to arbitrary RDF graphs (no syntactic restrictions) and which do not rely on satisfiability checking (are not bound by the principle of explosion).

Rule-based Reasoning Predating OWL 2—and in particular the provision of the OWL 2 RL/RDF ruleset—numerous rule-based entailment regimes were proposed in the literature to provide a partial axiomatisation of OWL's semantics. These regimes included Description Logic Programs (DLP) [26, 57], pD* [75, 76], RDFS-Plus [1], etc. Recognising the evident demand for rule-based support of OWL, in 2009, the W3C OWL Working Group standardised the OWL 2 RL profile and accompanying OWL 2 RL/RDF ruleset [24].The OWL 2 RL profile is a syntactic subset of OWL 2 which is implementable through translation to the Direct Semantics (DL-based semantics) or the RDF-Based Semantics (OWL 2 Full semantics). As such, the OWL 2 RL/RDF ruleset comprises a partial-axiomatisation of the OWL 2 RDF-Based Semantics which is applicable for arbitrary RDF graphs, and thus is is compatible with RDF Semantics [30]. We thus select OWL 2 RL/RDF as the most comprehensive, standard means of supporting RDFS *and* OWL entailment using rules, which largely subsumes the entailment possible through RDFS, DLP, pD*, RDFS-Plus, etc. For reference, we provide the OWL 2 RL/RDF ruleset in Appendix A, highlighting various characteristics which we will discuss as appropriate. (*Please also see [29] in these proceedings for discussion on the combination of rules and ontologies.*)

Forward Chaining We opt to perform *forward-chaining materialisation* of inferred data with respect to (a subset of) OWL 2 RL/RDF rules—i.e., we aim to make explicit the implicit data inferable through these rules (as opposed to, e.g., rewriting/extending queries and posing them against the original data in situ).

A materialisation approach offers two particular benefits:

- *pre-runtime execution*: materialisation can be conducted off-line (or, more accurately while loading data) avoiding the run-time expense of query-specific backward-chaining techniques which may adversely affect query response times;
- *consumer independent*: the inferred data provided by materialisation can subsequently be consumed in the same manner as explicit data, without the need for integrating a reasoning component into the runtime engine.

Note that in the spirit of *one size does not fit all*, forward-chaining materialisation is not a "magic-bullet": backward-chaining may be more propitious to support inferences where the amount of data involved is prohibitively expensive to materialise and index, and where these inferred data are infrequently required by the consumer application. Herein, we focus on materialisation, but grant that the inherent trade-off between offline forward-chaining and runtime backward-chaining warrants further investigation in another scope.

Alongside **scalability** and **robustness**, we identify two further requirements for our materialisation approach:

- **efficiency**: the reasoning algorithm must not only be able to process large-amounts of data, but should do so in as little computation time as possible;
- **terseness**: to reduce the burden on the consumer system—e.g., with respect to indexing or query-processing—we wish to keep a succinct volume of materialised data and aim instead for "reasonable" completeness.

Both of these additional requirements are intuitive, but also non-trivial, and so will provide important input for our design decisions.

OWL 2 RL/RDF Scalability Full materialisation with respect to the entire set of OWL 2 RL/RDF rules is infeasible for our use-case. First, a subset of OWL 2 RL/RDF rules are expressed informally—i.e., they are not formalised by means of Horn clauses—and may introduce new terms as a consequence of the rule, which in turn affects decidability (i.e., the achievability of a finite fixpoint). For example, OWL 2 RL/RDF rule dt-eq is specified as:

$$\forall \texttt{lt}_1, \texttt{lt}_2 \in \mathsf{L} \; with \; the \; same \; data \; value, \; infer \; (\texttt{lt}_1, \texttt{owl:sameAs}, \texttt{lt}_2) \, .$$

Note that this rule does not constrain $\texttt{lt}_1$ or $\texttt{lt}_2$ to be part of any graph or interpretation under analysis. Similarly, rule dt-diff entails pairwise `owl:differentFrom` relations between all literals with different data values, and rule dt-type2 entails an explicit membership triple for each literal to its datatype. These rules applied to, e.g., the value set of decimal-expressible real numbers (denotable by the datatype `xsd:decimal`) entail infinite triples.[17]

Aside from these datatype rules, the worst-case complexity of applying OWL 2 RL/RDF rules is cubic with respect to the known set of constants (a.k.a. the Herbrand universe); for example, consider the following two triples:

(`owl:sameAs`, `owl:sameAs`, `rdf:type`)
(`owl:sameAs`, `rdfs:domain`, `bad:Hub`)

Adding these two triples to any arbitrary RDF graph will lead to the inference of all possible (generalised) triples by the OWL 2 RL/RDF rules: i.e., the inference of $C \times C \times C$ (a.k.a. the Herbrand base), where $C \subset \mathsf{C}$ is the set of RDF constants (§ 2.1) mentioned in the OWL 2 RL/RDF ruleset and the graph (a.k.a. the Herbrand universe). The process involves the OWL 2 RL/RDF "equality rules" (eq-*) and the rule for supporting `rdfs:domain` (prp-dom), which lead to the inference of $|C|^3$ triples, as such *emulating* the explosive nature of inconsistency without actually requiring any inconsistency—we leave the details of the inferencing as an exercise for the reader (available in [34]).

This simple example raises concerns with respect to all of our defined requirements: materialising the required entailments for a large graph will be neither **scalable** nor **efficient**; even assuming that materialisation were possible, the result would not be **terse** (or be of any use at all to the consumer system); given that a single remote publisher can arbitrarily make such assertions in any location they like, such reasoning is clearly not **robust**.

Even for reasonable inputs, the result size and expense of OWL 2 RL/RDF materialisation can be prohibitive for our scenario. For example, chains of transitive relations of length n mandate quadratic ($\frac{n^2-n}{2}$) materialisation. Large equivalence classes (sets

[17] Typically, materialisation engines support non-standard versions of these rules using heuristics such as canonicalisation of datatype literals, or only applying the rules over literals that appear in the ruleset or in the data under analysis.

of individuals who are pairwise related by owl:sameAs) similarly mandate the materialisation of n^2 pairwise symmetric, reflexive and transitive owl:sameAs relations. Given our input sizes and the distinct possibility of such phenomena in our corpus, such quadratic materialisation quickly infringes upon our requirements for **scalability**, **efficiency** and arguably **terseness**.

Moreover, certain rules can materialise inferences which hold for every term in the graph—we call these inferences *tautological*. For example, the OWL 2 RL/RDF rule eq-ref materialises a reflexive owl:sameAs statement for every known term, reflecting the fact that everything is the same as itself. Thus, we omit such tautological rules (eq-ref for OWL 2 RL/RDF), viewing them as contrary to our requirement for **terseness**.

As such, the OWL 2 RL/RDF ruleset—and application thereof—requires significant tailoring to meet our requirements; we begin with our first non-standard optimisation in the following section.

2.3 Distinguishing Terminological Data

As previously described, RDFS/OWL allow for disseminating terminological data—loosely schema-level data—which provide definitions of classes and properties. Given a sufficiently large corpora collected from the Web, the percentage of terminological data is relatively small when compared to the volume of assertional data: typically—and as we will see in § 2.6—terminological data represent less than one percent of such a corpus [35, 37]. Assuming that the proportion of terminological data is quite small—and given that these data are among the most commonly accessed during reasoning—we formulate an approach around the assumption that such data can be efficiently handled and processed independently of the main bulk of assertional data. First, we provide some preliminaries relating to our notion of terminological data.

Meta-class We consider a *meta-class* as a class specifically of classes or properties; i.e., the members of a meta-class are themselves either classes or properties. *Herein, we restrict our notion of meta-classes to the set defined in RDF(S) and OWL specifications*, where examples include rdf:Property, rdfs:Class, owl:Class, owl:FunctionalProperty, owl:Restriction, owl:DatatypeProperty, etc.; note that rdfs:Resource, rdfs:Literal, e.g., are not considered meta-classes.

Meta-property A *meta-property* is one which has a meta-class as its domain; *again, we restrict our notion of meta-properties to the set defined in RDF(S) and OWL specifications*, where examples include rdfs:domain, rdfs:subClassOf, owl:hasKey, owl:inverseOf, owl:oneOf, owl:onProperty, owl:unionOf, etc.; note that rdf:type, owl:sameAs, rdfs:label, e.g., do *not* have a meta-class as domain and so are *not* considered meta-properties.

Terminological triple We define the set of *terminological triples* as the union of the following sets of triples:

1. triples with `rdf:type` as predicate and a meta-class as object;
2. triples with a meta-property as predicate;
3. triples forming a *valid* RDF list whose head is the object of a meta-property (e.g., a list used for `owl:unionOf`, `owl:intersectionOf`, etc.);

Our approach for separating terminological data is related to the area of partial evaluation and program specialisation of Logic Programs [50, 53, 44]: we take a generic (meta) program—such as RDFS, pD*, OWL 2 RL/RDF, etc.—and partially evaluate this program with respect to terminological knowledge. The result of this partial evaluation is a set of terminological inferences and a residual program which can be applied over the assertional data; this specialised *assertional program* is then primed using further optimisation before application over the bulk of assertional data.

Towards this goal, we begin by formalising the notion of a T-split rule which distinguishes between terminological and assertional atoms (T-atoms/A-atoms).

Definition 1 (T-split rule). *A* T-split rule R *is given as follows:*

$$H \leftarrow A_1, \ldots, A_n, T_1, \ldots, T_m \quad (n, m \geq 0)\,, \tag{2}$$

where the T_i, $0 \leq i \leq m$ atoms in the body (T-atoms) are all those that can only have terminological ground instances, whereas the A_i, $1 \leq i \leq n$ atoms (A-atoms), can have arbitrary ground instances. We use $\mathsf{TBody}(R)$ and $\mathsf{ABody}(R)$ to respectively denote the set of T-atoms and the set of A-atoms in the body of R.

Henceforth, we assume rules are T-split such that T-atoms and A-atoms can be referenced using the functions TBody and ABody when necessary.

Example 1. Let R_{EX} denote the following rule:

$$(\texttt{?x}, \texttt{a}, \texttt{?c2}) \leftarrow \underline{(\texttt{?c1}, \texttt{rdfs:subClassOf}, \texttt{?c2})}, (\texttt{?x}, \texttt{a}, \texttt{?c1})$$

When writing T-split rules, we denote $\mathsf{TBody}(R_{EX})$ by underlining: the underlined T-atom can only be bound by a triple with the meta-property `rdfs:subClassOf` as RDF predicate, and thus can only be bound by a terminological triple. The second atom in the body can be bound by assertional or terminological triples, and so is considered an A-atom. ◊

The notion of a *T-split program*—containing T-split rules and ground T-atoms and A-atoms—follows naturally. Distinguishing terminological atoms in rules enables us to define a form of stratified program execution, whereby a terminological fixpoint is reached first, and then the assertional data is reasoned over; we call this the *T-split least fixpoint*. Before we formalise this alternative fixpoint procedure, we must first describe our notion of a T-ground rule, where the variables appearing in T-atoms of a rule are grounded separately by terminological data:

Definition 2 (T-ground rule). *A* T-ground rule *is a set of rule instances for the T-split rule R given by grounding $\mathsf{TBody}(R)$ and the variables it contains across the rest of the rule. We denote the set of such rules for a program P and a set of facts I as $\mathsf{Ground}^T(P, I)$, defined as:*

$$\mathsf{Ground}^T(P, I) := \{\mathsf{Head}(R)\theta \leftarrow \mathsf{ABody}(R)\theta \mid R \in P,\, \exists I' \subseteq I \text{ s.t. } \theta = \mathsf{mgu}(\mathsf{TBody}(R), I')\}$$

The result is a set of rules whose T-atoms are grounded by the terminological data in I.

Example 2. Consider the T-split rule R_{EX} as before:

$$(\texttt{?x}, \texttt{a}, \texttt{?c2}) \leftarrow \underline{(\texttt{?c1}, \texttt{rdfs:subClassOf}, \texttt{?c2})}, (\texttt{?x}, \texttt{a}, \texttt{?c1})$$

Now let

$$I_{EX} \coloneqq \{\, (\texttt{foaf:Person}, \texttt{rdfs:subClassOf}, \texttt{foaf:Agent}), \\ (\texttt{foaf:Agent}, \texttt{rdfs:subClassOf}, \texttt{dc:Agent}) \,\}$$

Here,

$$\mathsf{Ground}^T(\{R_{EX}\}, I_{EX}) = \{\, (\texttt{?x}, \texttt{a}, \texttt{foaf:Agent}) \leftarrow (\texttt{?x}, \texttt{a}, \texttt{?foaf:Person}); \\ (\texttt{?x}, \texttt{a}, \texttt{dc:Agent}) \leftarrow (\texttt{?x}, \texttt{a}, \texttt{?foaf:Agent}) \,\}.$$

$\Diamond$

We can now formalise our notion of the T-split least fixpoint, where a terminological least model is determined, T-atoms of rules are grounded against this least model, and the remaining (proper) assertional rules are applied against the bulk of assertional data in the corpus. (In the following, we recall from § 2.1 the notions of the immediate consequence operator $\mathfrak{T}_P$, the least fixpoint $\mathsf{lfp}(\mathfrak{T}_P)$, and the least model $\mathsf{lm}(P)$ for a program P.)

Definition 3 (T-split least fixpoint). *The* T-split least fixpoint *for a program P is broken up into two parts: (i) the* terminological least fixpoint*, and (ii) the* assertional least fixpoint. *Let $P^F \coloneqq \{R \in P \mid \mathsf{Body}(R) = \emptyset\}$ be the set of facts in P,*[18] *let $P^{T\emptyset} \coloneqq \{R \in P \mid \mathsf{TBody}(R) \neq \emptyset, \mathsf{ABody}(R) = \emptyset\}$, let $P^{\emptyset A} \coloneqq \{R \in P \mid \mathsf{TBody}(R) = \emptyset, \mathsf{ABody}(R) \neq \emptyset\}$, and let $P^{TA} \coloneqq \{R \in P \mid \mathsf{TBody}(R) \neq \emptyset, \mathsf{ABody}(R) \neq \emptyset\}$. Note that $P = P^F \cup P^{T\emptyset} \cup P^{\emptyset A} \cup P^{TA}$. Now, let*

$$TP \coloneqq P^F \cup P^{T\emptyset}$$

denote the initial (terminological) program containing ground facts and T-atom only rules, and let $\mathsf{lm}(TP)$ denote the least model for the terminological program. Let

$$P^{A+} \coloneqq \mathsf{Ground}^T(P^{TA}, \mathsf{lm}(TP))$$

denote the set of (proper) rules achieved by grounding rules in P^{TA} with the terminological atoms in $\mathsf{lm}(TP)$: Now, let

$$AP \coloneqq \mathsf{lm}(TP) \cup P^{\emptyset A} \cup P^{A+}$$

denote the second (assertional) program containing all available facts and proper assertional rules. Finally, we can give the least model of the T-split program P as $\mathsf{lm}(AP)$ for AP derived from P as above—we more generally denote this by $\mathsf{lm}^T(P)$.

[18] Of course, P^F can refer to axiomatic facts and/or the initial facts given by an input knowledge-base.

An important question thereafter is how the standard fixpoint of the program $\mathsf{lm}(P)$ relates to the T-split fixpoint $\mathsf{lm}^T(P)$. Firstly, we show that the latter is *sound* with respect to the former:

Theorem 1 (T-split soundness). *For any program P, it holds that $\mathsf{lm}^T(P) \subseteq \mathsf{lm}(P)$.*

Proof available in [34].

Thus, for any given program containing rules and facts (as we define them), the T-split least fixpoint is necessarily a subset of the standard least fixpoint. Next, we look at characterising the *completeness* of the former with respect to the latter; beforehand, we need to define our notion of a T-Box:

Definition 4 (T-Box). *We define the* T-Box *of an interpretation I with respect to a program P as the subset of facts in I that are an instance of a T-atom of a rule in P:*

$$\mathsf{TBox}(P, I) := \{F \in I | \exists R \in P, \exists T \in \mathsf{TBody}(R) \text{ s.t. } T \triangleright F\} .$$

(Here we recall the $\triangleright$ notation of an instance [§ 2.1] whereby $A \triangleright B$ iff $\exists \theta$ s.t. $A\theta = B$.) Thus, our T-Box is precisely the set of terminological triples in a given interpretation (i.e., graph) that can be bound by a terminological atom of a rule in the program.

We now give a conditional proposition of completeness which states that if no new T-Box facts are produced during the execution of the assertional program, the T-split least model is equal to the standard least model.

Theorem 2 (T-split conditional completeness). *For any program P, its terminological program TP and its assertional program AP, if it holds that $\mathsf{TBox}(P, \mathsf{lm}(TP)) = \mathsf{TBox}(P, \mathsf{lm}(AP))$, then it holds that $\mathsf{lm}(P) = \mathsf{lm}^T(P)$.*

Corollary 1 (Rephrased condition for T-split completeness). *For any program P, if a rule with non-empty* ABody *does not infer a terminological fact, then $\mathsf{lm}(P) = \mathsf{lm}^T(P)$.*

Proofs available in [34].

So one may wonder when this condition of completeness is broken—i.e., when do rules with assertional atoms infer terminological facts? Analysis of how this can happen must be applied per rule-set, but for OWL 2 RL/RDF, we conjecture that such a scenario can only occur through (i) so called *non-standard use* of the set of RDFS/OWL meta-classes and meta-properties required by the rules, *or*, (ii) by the semantics of replacement for `owl:sameAs` (supported by OWL 2 RL/RDF rules eq-rep-*).[19]

We first discuss the effects of non-standard use for T-split reasoning over OWL 2 RL/RDF, starting with a definition.

Definition 5 (Non-standard triples). *With respect to a set of meta-properties MP and meta-classes MC, a* non-standard triple *is a terminological triple (T-fact wrt. MP/MC)* where additionally*:*

[19] We note that the phrase "non-standard use" has appeared elsewhere in the literature with the same intuition, but with slightly different formulation and intention; e.g., see [16].

– *a meta-class in MC appears in a position other than as the value of* `rdf:type`*;* or
– *a property in $MP \cup \{$*`rdf:type`*,*`rdf:first`*,*`rdf:rest`*$\}$ appears outside of the RDF predicate position.*

We call the set $MP \cup MC \cup \{$`rdf:type`,`rdf:first`,`rdf:rest`$\}$ the *restricted vocabulary*. (Note that restricting the use of `rdf:first` and `rdf:rest` would be superfluous for RDFS and pD* which do not support terminological axioms containing RDF lists.)

Now, before we formalise a proposition about the incompleteness caused by such usage, we provide an intuitive example thereof:

Example 3. As an example of incompleteness caused by non-standard use of the meta-property `owl:InverseFunctionalProperty`, consider:

$1_a.$ (`ex:KeyProperty`, `rdfs:subClassOf`, `owl:InverseFunctionalProperty`)
$2_a.$ (`ex:isbn13`, `a`, `ex:KeyProperty`)
$3_a.$ (`ex:The_Road`, `ex:isbn13`, `"978-0307265432"`)
$4_a.$ (`ex:Road%2C_The`, `ex:isbn13`, `"978-0307265432"`)

where triple (1_a) is considered non-standard use. The static T-Box in the terminological program will include the first triple, and, through the assertional rule cax-sco and triples (1_a) and (2_a) will infer:

$5_a.$ (`ex:isbn13`, `a`, `owl:InverseFunctionalProperty`)

but this T-fact will not be considered by the pre-ground T-atoms of the rules in the assertional program. Thus, the inferences:

$6_a.$ (`ex:The_Road`, `owl:sameAs`, `ex:Road%2C_The`)
$7_a.$ (`ex:Road%2C_The`, `owl:sameAs`, `ex:The_Road`)

which should hold through rule prp-ifp and triples (3_a), (4_a) and (5_a) will not be made.

A similar example follows for non-standard use of meta-classes; e.g.:

$1_b.$ (`ex:inSubFamily`, `rdfs:subClassOf`, `rdfs:subClassOf`)
$2_b.$ (`ex:Bos`, `ex:inSubFamily`, `ex:Bovinae`)
$3_b.$ (`ex:Daisy`, `a`, `ex:Bos`)

which through the assertional rule prp-spo1 and triples (1_b) and (2_b) will infer:

$4_b.$ (`ex:Bos`, `rdfs:subClassOf`, `ex:Bovinae`),

but not:

$5_b.$ (`ex:Daisy`, `a`, `ex:Bovinae`)

since triple (4_b) is not included in the terminological program. ◊

Theorem 3 (Conditional completeness for standard use). *Let $\mathcal{O}2\mathcal{R}'$ denote the set of (T-split) OWL 2 RL/RDF rules excluding* eq-rep-s*,* eq-rep-p *and* eq-rep-o*; let I be any interpretation not containing any non-standard use of the restricted vocabulary which contains (i) meta-classes or meta-properties appearing in the T-atoms of $\mathcal{O}2\mathcal{R}'$, and (ii)* `rdf:type`*,* `rdf:first`*,* `rdf:list`*; and let $P \coloneqq \mathcal{O}2\mathcal{R}' \cup I$; then, it holds that* $\mathsf{lm}(P) = \mathsf{lm}^T(P)$.

Sketch of proof involving inspection of OWL 2 RL/RDF rules available in [34].

Briefly, we note that [86] have given a similar result for RDFS by inspection of the rules, and that pD* inference relies on non-standard axiomatic triples whereby the above results do not translate naturally.

With respect to rules eq-rep-* (which we have thus far omitted), new terminological triples can be inferred from rules with non-empty ABody through the semantics of `owl:sameAs`, breaking the condition for completeness from Theorem 2. However, with respect to the T-split inferencing procedure, we conjecture that incompleteness can only be caused if `owl:sameAs` affects some constant in the TBody of an OWL 2 RL/RDF rule; we refer the interested reader to [34] for some examples and further discussion. In any case, note that (i) in our intended use-case, we do not apply rules `eq-rep-*` in our inferencing procedure due to scalability concerns—this will be discussed further in § 2.5; and (ii) we believe that in practice, T-split incompleteness through such `owl:sameAs` relations would only occur for rare corner cases. (For more detailed work looking at scalable "equality reasoning" for Linked Data, please see [34, § 7].)

Conceding the possibility of incompleteness—in particular in the presence of non-standard triples or `owl:sameAs` relations affecting certain terminological constants—we proceed by describing our implementation of the T-split program execution, how it enables unique optimisations, and how it can be used to derive a subset of OWL 2 RL/RDF rules which are linear with respect to assertional knowledge.

Implementing T-split Inferencing Given that the T-Box remains static during the application of the assertional program, our T-split algorithm enables a partial-indexing approach to reasoning, whereby only a subset of assertional triples—in particular those required by rules with multiple A-atoms in the body—need be indexed. Thus, the T-split closure can be achieved by means of two triple-by-triple scans of the corpus:

1. the first scan **identifies and separates out the T-Box and applies the terminological program**:
   (a) during the scan, any triples that are instances of a T-atom of a rule are indexed in memory;
   (b) after the scan, rules with only T-atoms in the body are applied over the in-memory T-Box until the terminological least model is reached, and rules with T-atoms and A-atoms in the body have their T-atoms grounded by these data;

(c) novel inferences in the terminological least model are written to an on-disk file (these will later be considered as part of the inferred output, and as input to the assertional program);

2. the second scan **applies the assertional program** over the main corpus and the terminological inferences;
   (a) each triple is individually checked to see if it unifies with an atom in an assertional rule body;
      i. if it unifies with a single-atom rule body, the inference is immediately applied;
      ii. if it unifies with a multi-atom rule body, the triple is indexed and the index is checked to determine whether the other atoms of the rule can be instantiated by previous triples—if so, the inference is applied;
   (b) inferred triples are immediately put back into step (2a), with an in-memory cache avoiding cycles and (partially) filtering duplicates.

The terminological program is applied using standard *semi-naïve evaluation* techniques, whereby only instances of rule bodies involving novel data will fire, ensuring that derivations are not needlessly and endlessly repeated (see, e.g., [80]).

We give a more formal break-down of the application of the assertional program in Algorithm 2.1. For our purposes, the A-Box input is the set of axiomatic statements in the rule fragment, the set of novel terminological inferences, and the entire corpus; i.e., we consider terminological data as also being assertional in a unidirectional form of punning [23].

First note that duplicate inference steps may be applied for rules with only one atom in the body (Lines 11–14): one of the main optimisations of our approach is that it minimises the amount of data that we need to index, where we only wish to store triples which may be necessary for later inference, and where triples *only* grounding single atom rule bodies need not be indexed. To provide *partial* duplicate removal, we instead use a Least-Recently-Used (LRU) cache over a sliding window of recently encountered triples (Lines 7 & 8)—outside of this window, we may not know whether a triple has been encountered before or not, and may repeat inferencing steps.

Thus, in this *partial-indexing* approach, we need only index those triples which are matched by a rule with a multi-atom body (Lines 15–25). For indexed triples, aside from the LRU cache, we can additionally check to see if that triple has been indexed before (Line 19) and we can apply a semi-naïve check to ensure that we only materialise inferences which involve the current triple (Line 20). We note that as the assertional index is required to store more data, the two-scan approach becomes more inefficient than the "full-indexing" approach; in particular, a rule with a body atom containing all variable terms will require indexing of all data, negating the benefits of the approach; e.g., if the rule OWL 2 RL/RDF rule **eq-rep-s**:

$$(?s', ?p, ?o) \leftarrow (?s, \texttt{owl:sameAs}, ?s'),(?s, ?p, ?o)$$

is included in the assertional program, the entire corpus of assertional data must be indexed (in this case according to subject) because of the latter "open" atom. We emphasise that our partial-indexing performs well if the assertional index remains small and performs best if every proper rule in the assertional program has only one A-atom

Algorithm 2.1. Reason over the A-Box

Require: ABOX: $\mathtt{A}$ /* $\{t_0 \dots t_m\}$ */
Require: ASSERTIONAL PROGRAM: AP /* $\{R_0 \dots R_n\}, \mathsf{TBody}(R_i) = \emptyset$ */
Index $\coloneqq \{\}$ /* *triple index* */
LRU $\coloneqq \{\}$ /* *fixed-size, least recently used cache* */
for all $t \in \mathtt{A}$ **do**
  $G_0 \coloneqq \{\}, G_1 \coloneqq \{t\}, i \coloneqq 1$
  while $G_i \neq G_{i-1}$ **do**
    for all $t_\delta \in G_i \setminus G_{i-1}$ **do**
      if $t_\delta \notin$ LRU **then** /* *if* $t_\delta \in$ LRU, *make* t_δ *most recent entry* */
        add t_δ to LRU /* *remove eldest entry if necessary* */
        $\mathtt{output}(t_\delta)$
        for all $R \in AP$ **do**
          if $|\mathsf{Body}(R)| = 1$ **then**
            if $\exists\theta$ s.t. $\{t_\delta\} = \mathsf{Body}(R)\theta$ **then**
              $G_{i+1} \coloneqq G_{i+1} \cup \mathsf{Head}(R)\theta$
            end if
          else
            if $\exists\theta$ s.t. $t_\delta \in \mathsf{Body}(R)\theta$ **then**
              $card = |\text{Index}|$
              Index $\coloneqq$ Index $\cup \{t_\delta\}$
              if $card \neq |\text{Index}|$ **then**
                for all θ s.t. $\mathsf{Body}(R\theta) \subseteq \text{Index}, t_\delta \in \mathsf{Body}(R\theta)$ **do**
                  $G_{i+1} \coloneqq G_{i+1} \cup \mathsf{Head}(R\theta)$
                end for
              end if
            end if
          end if
        end for
      end if
    end for
    i++
    $G_{i+1} \coloneqq$ copy(G_i) /* *copy inferences to new set to avoid cycles* */
  end while
end for
return $\mathtt{output}$ /* *on-disk inferences* */

in the body—in the latter case, no assertional indexing is required. We will use this observation to identify a subset of T-split OWL 2 RL/RDF rules which are linear with respect to the assertional knowledge in § 2.5, but first we look at some generic optimisations for the assertional program.

2.4 Optimising the Assertional Program

Note that in Algorithm 2.1 Line 10, all rules are checked for all triples to see if an inference should take place. Given that (i) the assertional program will be applied over a corpus containing in the order of a billion triples; (ii) the process of grounding the T-atoms of T-split rules may lead to a large volume of assertional rules given a sufficiently complex terminology; we deem it worthwhile to investigate some means of optimising the execution of the assertional program. Herein, we discuss such optimisations and provide initial evaluation thereof—note that since our assertional program contains only assertional atoms, we herein omit the T-split notation where $\mathsf{Body}(R)$ always refers to a purely assertional body.

Merging Equivalent T-ground Rules Applying the T-grounding of rules to derive purely assertional rules may generate "equivalent rules": rules which can be unified by an *bijective variable rewriting*. Similarly, there may exist T-ground rules with "equivalent bodies" which can be *merged* into one rule. To formalise these notions, we first define the bijective variable rewriting function used to determine equivalence of atoms.

Definition 6 (Variable rewriting). *A bijective* variable rewriting function *is an automorphism on the set of variables, given simply as:*

$$\nu : \mathsf{V} \mapsto \mathsf{V}$$

As such, this function is a specific form of variable substitution, where two atoms which are unifiable by such a rewriting are considered *equivalent*:

Definition 7 (Equivalent atoms). *Two atoms are* equivalent *(denoted $A_1 \diamond\!\!\triangleright A_2$ reflecting the fact that both atoms are instances of each other) iff they are unifiable by a bijective variable rewriting:*[20]

$$A_1 \diamond\!\!\triangleright A_2 \Leftrightarrow \exists \nu \text{ s.t. } A_1\nu = A_2$$

Equivalence of a set of atoms follows naturally. Two rules are body-equivalent *($R_1 \diamond\!\!\triangleright_{\mathrm{b}} R_2$) iff their bodies are equivalent:*

$$R_1 \diamond\!\!\triangleright_{\mathrm{b}} R_2 \Leftrightarrow \mathsf{Body}(R_1) \diamond\!\!\triangleright \mathsf{Body}(R_2) \Leftrightarrow \exists \nu \text{ s.t. } \mathsf{Body}(R_1)\nu = \mathsf{Body}(R_2)$$

Two rules are considered fully-equivalent *if their bodies and heads are unifiable by the same variable rewriting:*

$$R_1 \diamond\!\!\triangleright_{\mathrm{r}} R_2 \Leftrightarrow \exists \nu \text{ s.t. } \Big(\mathsf{Body}(R_1)\nu = \mathsf{Body}(R_2) \wedge \mathsf{Head}(R_1)\nu = \mathsf{Head}(R_2)\Big)$$

[20] Note that in the unification, only the variables in the left atom are rewritten and not both; otherwise two atoms such as (`?a`, `foaf:knows`, `?b`) and (`?b`, `foaf:knows`, `?c`) would not be equivalent: they could not be aligned by any (necessarily *injective*) rewriting function ν.

Note that fully-equivalent rules are considered redundant, and all but one can be removed without affecting the computation of the least model. Using these equivalence relations, we can now define our *rule-merge* function (again recall from § 2.1 our interpretation of multi-atom heads as being conjunctive, and a convenient representation of the equivalent set of rules):

Definition 8 (Rule merging). *Given an equivalence class of rules* $[R]_{\triangleleft\!\triangleright_{\mathrm{b}}}$*—a set of rules between which* $\triangleleft\!\triangleright_{\mathrm{b}}$ *holds—select a* canonical rule $R \in [R]_{\triangleleft\!\triangleright_{\mathrm{b}}}$*; we can now describe the* rule-merge *of the equivalence class as*

$$\mathsf{merge}([R]_{\triangleleft\!\triangleright_{\mathrm{b}}}) \coloneqq \mathsf{Head}_{[R]_{\triangleleft\!\triangleright_{\mathrm{b}}}} \leftarrow \mathsf{Body}(R)$$

where

$$\mathsf{Head}_{[R]_{\triangleleft\!\triangleright_{\mathrm{b}}}} \coloneqq \bigcup_{R_i \in [R]_{\triangleleft\!\triangleright_{\mathrm{b}}}} \mathsf{Head}(R_i)\nu_i \text{ s.t. } \mathsf{Body}(R_i)\nu_i = \mathsf{Body}(R)$$

Now take a program P *and let:*

$$P/{\triangleleft\!\triangleright_{\mathrm{b}}} \coloneqq \{[R]_{\triangleleft\!\triangleright_{\mathrm{b}}} \mid R \in P\}$$

denote the quotient set of P *given by* $\triangleleft\!\triangleright_{\mathrm{b}}$*: the set of all equivalent classes* $[R]_{\triangleleft\!\triangleright_{\mathrm{b}}}$ *wrt. the equivalence relation* $\triangleleft\!\triangleright_{\mathrm{b}}$ *in* P*. We can generalise the* rule merge function *for a set of rules as*

$$\begin{aligned} \mathsf{merge}: \ & 2^{\mathsf{Rules}} \to 2^{\mathsf{Rules}} \\ & P \mapsto \bigcup_{[R]_{\triangleleft\!\triangleright_{\mathrm{b}}} \in P/{\triangleleft\!\triangleright_{\mathrm{b}}}} \mathsf{merge}([R]_{\triangleleft\!\triangleright_{\mathrm{b}}}) \end{aligned}$$

Example 4. Take three T-ground rules:

$$\begin{aligned} (\texttt{?x}, \texttt{a}, \texttt{foaf:Person}) &\leftarrow (\texttt{?x}, \texttt{foaf:img}, \texttt{?y}) \\ (\texttt{?s}, \texttt{foaf:depicts}, \texttt{?o}) &\leftarrow (\texttt{?s}, \texttt{foaf:img}, \texttt{?o}) \\ (\texttt{?a}, \texttt{foaf:depicts}, \texttt{?b}) &\leftarrow (\texttt{?a}, \texttt{foaf:img}, \texttt{?b}) \end{aligned}$$

The second rule can be merged with the first using $\nu_1 = \{\texttt{?s/?x}, \texttt{?o/?y}\}$, which gives:

$$(\texttt{?x}, \texttt{a}, \texttt{foaf:Person}),(\texttt{?x}, \texttt{foaf:depicts}, \texttt{?y}) \leftarrow (\texttt{?x}, \texttt{foaf:img}, \texttt{?y})$$

The third rule can be merged with the above rule using $\nu_1 = \{\texttt{?a/?x}, \texttt{?b/?y}\}$ to give:

$$(\texttt{?x}, \texttt{a}, \texttt{foaf:Person}),(\texttt{?x}, \texttt{foaf:depicts}, \texttt{?y}) \leftarrow (\texttt{?x}, \texttt{foaf:img}, \texttt{?y})$$

...the same rule. This demonstrates that the merge function removes redundant fully-equivalent rules. ◊

Merging the rules thus removes redundant rules, and reduces the total number of rule applications required for each triple without affecting the final least model:

Proposition 1. *For any program* P*,* $\mathsf{lm}(P) = \mathsf{lm}(\mathsf{merge}(P))$*.*

Sketch of proof available in [34].

Rule Index We have reduced the amount of rules in the assertional program through merging; however, given a sufficiently complex T-Box, we may still have a prohibitive number of rules for efficient recursive application. We now look at the use of a rule index which maps a fact to rules containing a body atom for which that fact is an instance, thus enabling the efficient identification and application of only relevant rules for a given triple.

Definition 9 (Rule lookup). *Given a fact F and program P, the* rule lookup function *returns all rules in the program containing a body atom for which F is an instance:*

$$\begin{aligned}\mathsf{lookup}: \ \mathsf{Facts} \times 2^{\mathsf{Rules}} &\rightarrow 2^{\mathsf{Rules}} \\ (F, P) &\mapsto \left\{ R \in P \mid \exists B_i \in \mathsf{Body}(R) \text{ s.t. } B_i \triangleright F \right\}\end{aligned}$$

Now, instead of attempting to apply all rules, for each triple we can perform the above lookup function and return only triples from the assertional program which could potentially lead to a successful rule application.

Example 5. Given a triple:

$$t \coloneqq (\texttt{ex:me}, \texttt{a}, \texttt{foaf:Person})$$

and a simple example ruleset:

$$\begin{aligned}P \coloneqq \{&(\texttt{?x}, \texttt{a}, \texttt{foaf:Person}) \leftarrow (\texttt{?x}, \texttt{foaf:img}, \texttt{?y}), \\ &(\texttt{?x}, \texttt{a}, \texttt{foaf:Agent}) \leftarrow (\texttt{?x}, \texttt{a}, \texttt{foaf:Person}), \\ &(\texttt{?y}, \texttt{a}, \texttt{rdfs:Class}) \leftarrow (\texttt{?x}, \texttt{a}, \texttt{?y})\}\end{aligned}$$

$\mathsf{lookup}(t, P)$ returns a set containing the latter two rules. ◊

With respect to implementing this lookup function, we require a rule index. A triple pattern has $2^3 = 8$ possible forms: $(?, ?, ?)$, $(s, ?, ?)$, $(?, p, ?)$, $(?, ?, o)$, $(s, p, ?)$, $(?, p, o)$, $(s, ?, o)$, (s, p, o). Thus, we require eight indices for indexing body patterns, and eight lookups to perform $\mathsf{lookup}(t, P)$ and find all relevant rules for a triple. We use seven in-memory hashtables storing the constants of the rule antecedent patterns as key, and a set of rules containing such a pattern as value; e.g., $\{(\texttt{?x}, \texttt{a}, \texttt{foaf:Person})\}$ is put into the $(?, p, o)$ index with (a,foaf:Person) as key. Rules containing $(?, ?, ?)$ patterns without constants are stored in a set, as they are relevant to all triples—they are returned for all lookups.

We further optimise the rule index by linking dependencies between rules, such that once one rule fires, we can determine which rules should fire next without requiring an additional lookup. This is related to the notion of a rule graph in Logic Programming (see, e.g., [67]):

Definition 10 (Rule graph). *A* rule graph *is defined as a directed graph:*

$$\Gamma \coloneqq (P, \hookrightarrow)$$

such that:[21]

$$R_i \hookrightarrow R_j \Leftrightarrow \exists B \in \mathsf{Body}(R_j), \exists H \in \mathsf{Head}(R_i) \text{ s.t. } B \triangleright H$$

where $R_i \hookrightarrow R_j$ is read as "R_j follows R_i".

By building and encoding such a rule graph into our index, we can "wire" the recursive application of rules for the assertional program. However, from the merge function (or otherwise) there may exist rules with large sets of head atoms. We therefore extend the notion of the rule graph to a directed labelled graph with the inclusion of a labelling function

Definition 11 (Rule-graph labelling). *Let Λ denote a labelling function as follows:*

$$\begin{aligned} \Lambda : \ & \mathsf{Rules} \times \mathsf{Rules} \to 2^{\mathsf{Atoms}} \\ & (R_i, R_j) \mapsto \left\{ H \in \mathsf{Head}(R_i) \mid \exists B \in \mathsf{Body}(R_j) \text{ s.t. } B \triangleright H \right\} \end{aligned}$$

A labelled rule graph *is thereafter defined as a directed labelled graph:*

$$\Gamma^{\Lambda} \coloneqq (P, \hookrightarrow, \Lambda)$$

Each edge in the rule graph is labelled with $\Lambda(R_i, R_j)$, denoting the set of atoms in the head of R_i that, when grounded, would be matched by atoms in the body of R_j.

Example 6. Take the two rules:

R_i. (y, a, foaf:Image),(?x, a, foaf:Person) ← (?x, foaf:img, ?y)
R_j. (s, a, foaf:Agent) ← (?s, a, foaf:Person)

We say that $R_i \overset{\lambda}{\hookrightarrow} R_j$, where $\lambda = \Lambda(R_i, R_j) = \{(?x, a, foaf:Person)\}$. ◊

In practice, our rule index stores sets of elements of a linked list, where each element contains a rule and links to rules which are relevant for the atoms in that rule's head. Thus, for each input triple, we can retrieve all relevant rules for all eight possible patterns, apply those rules, and if successful, follow the respective labelled links to recursively find relevant rules without re-accessing the index until the next input triple.

Rule Saturation We briefly describe the final optimisation technique we investigated, but which later evaluation demonstrated to be mostly disadvantageous: *rule saturation*. We say that a subset of dependencies in the rule graph are *strong dependencies*, where the successful application of one rule *will* always lead to the successful application of another. Now, we can saturate rules with single-atom bodies by pre-computing the recursive rule application of its dependencies; we give the gist with an example:

Example 7. Take rules

[21] Here, we recall from § 2.1 the '▷' notation for an instance.

R_i. (?x, a, foaf:Person),(?y, a, foaf:Image) ← (?x, foaf:img, ?y)
R_j. (?s, a, foaf:Agent) ← (?s, a, foaf:Person)
R_k. (?y, a, rdfs:Class) ← (?x, a, ?y)

We can see that $R_i \hookrightarrow R_j$, $R_i \hookrightarrow R_k$, $R_j \hookrightarrow R_k$ as before. Now, we can remove the links from R_i to R_j and R_k by saturating R_i to:

R'_i. (?x, a, foaf:Person),(?y, a, foaf:Image),(?x, a, foaf:Agent),
(foaf:Person, a, rdfs:Class),(foaf:Image, a, rdfs:Class),
(foaf:Agent, a, rdfs:Class) ← (?x, foaf:img, ?y)

and, analogously, we can remove the links from R_j to R_k by saturating R_j to:

R'_j. (?s, a, foaf:Agent),(foaf:Agent, a, rdfs:Class) ← (?s, a, foaf:Person)

Thus, the index now stores R'_i, R'_j, R_k, but without the links between them. ◊

However, as we will see in § 2.4, our empirical analysis found rule saturation to be mostly disadvantageous: although it decreases the number of necessary rule applications, as a side-effect, saturated rules can immediately produce a large batch of duplicates which would otherwise have halted a traversal of the rule graph early on. Using the above example, consider encountering the following sequence of input triples:

1. (ex:Fred, a, foaf:Person)
2. (ex:Fred, foaf:img, ex:FredsPic)

The first triple will fire rule R'_j and R_k; the second triple will subsequently fire rule R'_i, and in so doing, will produce a superset of inferences already given by its predecessor. Without saturation, the second triple would fire R_i, identify (ex:Fred, a, foaf:Person) as a duplicate, and instead only fire R_k for (ex:FredsPic, a, foaf:Image).

Preliminary Performance Evaluation We now perform some (relatively) small-scale experiments to empirically (in)validate our optimisations for the assertional program execution. Experiments are run on a 2.2GHz Opteron x86-64, 4GB main memory, 160GB SATA hard-disks, running Java 1.6.0_12 on Debian 5.0.4.

We applied reasoning for RDFS (minus the infinite rdf:_n axiomatic triples [30]), pD* and OWL 2 RL/RDF over LUBM(10) [27], consisting of about 1.27 million assertional triples and 295 terminological triples.[22] For each rule profile, we applied the following configurations:

1. N: **no partial evaluation**: T-Box atoms are bound at runtime from an in-memory triple-store;

[22] Note that we exclude lg/gl rules for RDFS/pD* since we allow generalised triples [24]. We also restrict OWL 2 RL/RDF datatype reasoning to apply only to literals in the program.

2. NI: **no partial evaluation with linked (meta-)rule index**;
3. P: **partial-evaluation**: generating and applying an assertional program;
4. PI: **partial evaluation with linked rule index**;
5. PIM: **partial evaluation with linked rule index and rule merging**;
6. PIMS: **partial evaluation with linked rule index, rule merging and rule saturation**.

Table 1 enumerates the results for each profile, with a breakdown of (i) the number of inferences made, (ii) the total number of assertional rules generated, (iii) the total number of merged rules; and for each of the six configurations; (iv) the time taken, (v) the total number of attempted rule applications—i.e., the total number of times a triple is *checked* to see if it grounds a body atom of a rule to produce inferences—and the percent of rule applications which generated inferences, and (vi) the number of duplicate triples filtered out by the LRU cache (Lines 7 & 8, Algorithm 2.1).

In all approaches, applying the non-optimised partially evaluated (assertional) program takes the longest: although the partially evaluated rules are more efficient to apply, this approach requires an order of magnitude more rule applications than directly applying the meta-program, and so applying the unoptimised residual assertional program takes approximately $2\times$ to $4\times$ longer than the baseline.

With respect to rule indexing, the technique has little effect when applying the meta-program directly—many of the rules contain open patterns in the body. Although the number of rule applications diminishes somewhat, the expense of maintaining and accessing the rule index actually worsens performance by between 10% and 20%. However, with the partially evaluated rules, more variables are bound in the body of the rules, and thus triple patterns offer more selectivity and, on average, the index returns fewer rules. We see that for PI and for each profile respectively, the rule index sees a 78%, 69% and 72% reduction in the equivalent runtime (P) without the rule index; the reduction in rule applications (73%, 80%, 86% reduction resp.) is significant enough to more than offset the expense of maintaining and using the index. With respect to the baseline (N), PI makes a 10%, 38% and 45% saving respectively; notably, for RDFS, the gain in performance over the baseline is less pronounced, where, relative to the more complex rulesets, the number of rule applications is not signficantly reduced by partial evaluation and indexing.

Merging rules provided a modest saving across all rulesets, with PIM giving a 9%, 3% and 6.5% saving in runtime and a 12%, 8% and 4% saving in rule applications over PI respectively for each profile. Note that although OWL 2 RL/RDF initially creates more residual rules than pD* due to expanded T-Box level reasoning, these are merged to a number just above pD*: OWL 2 RL supports intersection-of inferencing used by LUBM and not in pD*. LUBM does not contain OWL 2 constructs, but redundant meta-rules are factored out during the partial evaluation phase.

Finally, we look at the effect of saturation and approach PIMS. For RDFS, we encountered a 15% reduction in runtime over PIM, with a 21% reduction in rule applications required. However, for pD* we encountered a 2% *increase* in runtime over that of PIM despite a 34% reduction in rule applications: as previously alluded to, the cache was burdened with $2.6\times$ more duplicates, negating the benefits of fewer rule applications. Similarly, for OWL 2 RL/RDF, we encountered a 4% increase in runtime over

RDFS						
inferred	0.748 million					
T-ground rules	149					
after merge	87					
config.	N	NI	P	PI	PIM	PIMS
time (s)	99	117	404	89	81	**69**
rule apps (m)	16.5	15.5	308	11.3	9.9	**7.8**
% success	43.4	46.5	2.4	**64.2**	62.6	52.3
cache hits (m)	10.8	10.8	8.2	8.2	8.2	**8.1**

pD*						
inferred	1.328 million					
T-ground rules	175					
after merge	108					
config.	N	NI	P	PI	PIM	PIMS
time (s)	365	391	734	227	**221**	225
rule apps (m)	62.5	50	468	22.9	21.1	**13.9**
% success	18.8	23.4	2.6	51.5	48.7	**61.3**
cache hits (m)	19.1	19.1	15.1	15.1	**14.9**	38.7

OWL 2 RL/RDF						
inferred	1.597 million					
T-ground rules	378					
after merge	119					
config.	N	NI	P	PI	PIM	PIMS
time (s)	858	940	1,690	474	**443**	465
rule apps (m)	149	110	1,115	81.8	78.6	**75.6**
% success	4.2	5.6	0.8	10.5	6.8	**15**
cache hits (m)	16.5	16.5	13.1	13	**12.7**	34.4

Table 1. Details of reasoning for LUBM(10)—containing 1.27M assertional triples and 295 terminological triples—given different reasoning configurations (the most favourable result for each row is highlighted in bold)

that of PIM despite a 4% reduction in rule applications: again, the cache encountered 2.7× more duplicates.

The purpose of this evaluation is to give a granular analysis and empirical justification for our optimisations for different rule-based profiles: one might consider different scenarios (such as a terminology-heavy corpus) within which our optimisations may not work. However, we will later demonstrate these optimisations—with the exception of rule saturation—to be propitious for our scenario of reasoning over Linked Data.

It is worth noting that—aside from reading input and writing output—we performed the above experiments almost entirely in-memory. Given the presence of (pure) assertional rules which have multi-atom bodies where one such atom is "open" (all terms are

variables)—viz., pD* rule rdfp11 and OWL 2 RL/RDF rules eq-rep-*—we currently must naïvely store *all* data in memory, and cannot scale much beyond LUBM(10).[23]

2.5 Towards Linked Data Reasoning

With the notions of a T-split program, partial evaluation and assertional program optimisations in hand, we now reunite with our original use-case of Linked Data reasoning, for which we move our focus from clean corpora in the order of a million statements to our corpus in the order of a billion statements collected from almost four million sources—we will thus describe some trade-offs we make in order to shift up (at least) these three orders of magnitude in scale, and to be tolerant to noise and impudent data present in the corpus. More specifically, we:

1. first describe, motivate and characterise the scalable subset of OWL 2 RL/RDF that we implement based partially on the discussion in the previous section;
2. introduce and describe *authoritative reasoning*, whereby we include cautious consideration of the source of terminology into the reasoning process;
3. outline our distribution strategy for reasoning;
4. evaluate our methods by applying reasoning over our Linked Data evaulation corpus of 1.12 billion quadruples crawled from 4 million RDF/XML documents.

"A-linear" OWL 2 RL/RDF Again, for a generic set of RDF rules (which do not create new terms in the head), the worst case complexity is cubic—in § 2.2 we have already demonstrated a simple example which instigates cubic reasoning for OWL 2 RL/RDF rules, and discussed how, for many reasonable inputs, rule application is quadratic. Given our use-case, we want to define a profile of rules which will provide linear complexity with respect to the assertional data in the corpus: what we call "A-linearity".

In fact, in the field of Logic Programming (and in particular Datalog) the notion of a linear program refers to one which contains rules with no more than one recursive atom in the body—a recursive atom being one which cannot be instantiated from an inference (e.g., see [15]).[24] For Datalog, recursiveness is typically defined on the level of predicates using the notion of *intensional predicates*, which represent facts that can (only) be inferred by the program, and *extensional predicates*, which represent facts in the original data; atoms with intensional predicates are non-recursive [15]. Since we deal with a single ternary predicate, such a predicate-level distinction does not apply, but the general notion of recursiveness does. This has a notable relationship to our distinction of terminological knowledge—which we deem to be recursive only within itself (assuming standard use of the meta-vocabulary and "well-behaved equality" involving `owl:sameAs`)—and assertional knowledge which *is* recursive.

[23] We could consider storing data in an on-disk index with in-memory caching; however, given the morphology and volume of the assertional data, and the frequency of lookups required, we believe that the cache hit rate would be low, and that the naïve performance of the on-disk index would suffer heavily from hard-disk latency, becoming a severe bottleneck for the reasoner.

[24] There is no relation between a linear program in our case, and the field of Linear Programming [83].

Based on these observations, we identify an A-linear subset of OWL 2 RL/RDF rules which contain only one recursive/assertional atom in the body, and apply only these rules. Taking this subset as our "meta-program", after applying our T-grounding of meta-rules during partial evaluation, the result will be a set of facts and proper rules with only one assertional atom in the body. The resulting linear assertional program can then be applied without any need to index the assertional data (other than for the LRU duplicates soft-cache); also, since we do not need to compute assertional *joins*—i.e., to find the most general unifier of multiple A-atoms in the data—we can employ a straightforward distribution strategy for applying the program.

Definition 12 (A-linear program). *Let P be any T-split (a.k.a. meta) program. We denote the* A-linear program *of P by $P^{\propto\mathsf{A}}$ defined as follows:*

$$P^{\propto\mathsf{A}} := \{R \in P : |\mathsf{ABody}(R)| \leq 1\}$$

(Note that by the above definition, $P^{\propto\mathsf{A}}$ also includes the pure-terminological rules and the facts of P.)

Thus, the proper rules of the assertional program $AP^{\propto\mathsf{A}}$ generated from an A-linear meta-program $P^{\propto\mathsf{A}}$ will only contain one atom in the head. For convenience, we denote the A-linear subset of OWL 2 RL/RDF by $\mathcal{O}2\mathcal{R}^{\propto\mathsf{A}}$, which consists of rules in Tables 13–16 (Appendix A).

Thereafter, the assertional program demonstrates two important characteristics with respect to scalability: (i) the assertional program can be independently applied over subsets of the assertional data, where a subsequent union of the resultant least models will represent the least model achievable by application of the program over the data in whole; (ii) the volume of materialised data and the computational expense of applying the assertional program are linear with respect to the assertional data.

Proposition 2 (Assertional partitionability). *Let I be any interpretation, and $\{I_1, \ldots, I_n\}$ be any set of interpretations such that:*

$$I = \bigcup_{i=1}^{n} I_i$$

Now, for any meta-program P, its A-linear subset $P^{\propto\mathsf{A}}$, and the assertional program $AP^{\propto\mathsf{A}}$ derived therefrom, it holds that:

$$\mathsf{lm}(AP^{\propto\mathsf{A}} \cup I) = \bigcup_{i=1}^{n} \mathsf{lm}(AP^{\propto\mathsf{A}} \cup I_i)$$

Proof. (Sketch) Follows naturally from the fact that rules in $AP^{\propto\mathsf{A}}$ (i) are monotonic and (ii) only contain single-atom bodies. □

Thus, deriving the least model of the assertional program can be performed over any partition of an interpretation; the set union of the resultant least models is equivalent to the least model of the unpartitioned interpretation. Aside from providing a straightforward distribution strategy, this result allows us to derive an upper-bound on the cardinality of the least model of an assertional program.

Proposition 3 (A-Linear least model size). *Let $AP^{\propto\mathsf{A}}$ denote any A-linear assertional program composed of RDF proper rules and RDF facts composed of ternary-arity atoms with the ternary predicate T. Further, let $I^{\propto\mathsf{A}}$ denote the set of facts in the program and $PR^{\propto\mathsf{A}}$ denote the set of proper rules in the program (here, $AP^{\propto\mathsf{A}} = I^{\propto\mathsf{A}} \cup PR^{\propto\mathsf{A}}$). Also, let the function* Const *denote the Herbrand universe of a set of atoms (the set of RDF constants therein), and let τ denote the cardinality of the Herbrand universe of the heads of all rules in $PR^{\propto\mathsf{A}}$ (the set of RDF constants in the heads of the proper T-ground rules of $AP^{\propto\mathsf{A}}$) as follows:*

$$\tau = \left|\mathsf{Const}\Big(\bigcup_{R \in PR^{\propto\mathsf{A}}} \mathsf{Head}(R)\Big)\right|$$

Finally, let α denote the cardinality of the set of facts:

$$\alpha = |I^{\propto\mathsf{A}}|$$

Then it holds that:

$$|\mathsf{Im}(AP^{\propto\mathsf{A}})| \leq \tau^3 + \alpha(9\tau^2 + 27\tau + 27)$$

Proof given in [34].

Note that τ is given by the terminology (more accurately the T-Box) of the data and the terms in the heads of the original meta-program. Considering τ as a constant, we arrive at the maximum size of the least model as $c + c\alpha$: i.e., the least model is linear with respect to the assertional data. In terms of rule applications, the number of rules is again a function of the terminology and meta-program, and the maximum number of rule applications is the product of the number of rules (considered a constant) and the maximum size of the least model. Thus, the number of rule applications remains linear with respect to the assertional data. This is a tenuous result with respect to scalability, and constitutes a refactoring of the cubic complexity to separate out a static terminology. Thereafter, assuming the terminology to be small, the constant c will be small and the least model will be **terse**; however, for a sufficiently complex terminology, obviously the τ^3 and $\alpha\tau^2$ factors begin to dominate—for a terminology heavy program, the worst-case complexity again approaches τ^3. Thus, applying an A-linear subset of a program is again not a "magic bullet" for scalability, although it should demonstrate scalable behaviour for small terminologies (i.e., where τ is small) and/or other reasonable inputs.

Moving forward, we select an A-linear subset of the OWL 2 RL/RDF ruleset for application over our ruleset. This subset is enumerated in Appendix A, with rule tables categorised by terminological and assertional arity of rule bodies. Again, we also make some other amendments to the ruleset:

1. we omit datatype rules which lead to the inference of (near-)infinite triples;
2. we omit inconsistency checking rules;
3. for reasons of **terseness**, we omit rules which infer 'tautologies'—statements that hold for every term in the graph, such as reflexive `owl:sameAs` statements (we also filter these from the output).

Authoritative Reasoning In preliminary evaluation of our Linked Data reasoning [35], we encountered a puzzling deluge of inferences: We found that remote documents sometimes cross-define terms resident in popular vocabularies, changing the inferences *authoritatively* mandated for those terms. For example, we found one document[25] which defines `owl:Thing` to be an element (i.e., a subclass) of 55 union class descriptions—thus, materialisation wrt. OWL 2 RL/RDF rule **cls-uni** [24, Table 6] over any member of `owl:Thing` would infer 55 additional memberships for these obscure union classes. We found another document[26] which defines nine *properties* as the domain of `rdf:type`—again, anything defined to be a member of any class would be inferred to be a member of these nine *properties* by rules prp-dom. Even aside from "cross-defining" core RDF(S)/OWL terms, popular vocabularies such as FOAF were also affected (we will see more in the evaluation presented in § 2.6).

In order to curtail the possible side-effects of open Web data publishing (as also exemplified by the two triples which cause cubic reasoning in § 2.2), we include the source of data in inferencing. Our methods are based on the view that a publisher instantiating a vocabulary's term (class/property) thereby accepts the inferencing mandated by that vocabulary (and recursively referenced vocabularies) for that term. Thus, once a publisher instantiates a term from a vocabulary, only that vocabulary and its references should influence what inferences are possible through that instantiation. As such, we ignore unvetted terminology at the potential cost of discounting serendipitous mappings provided by independent parties, since we currently have no means of distinguishing "good" third-party contributions from "bad" third-party contributions. We call this more conservative form of reasoning *authoritative reasoning*, which only considers authoritatively published terminological data, and which we now describe. (*Please also see [77] in these proceedings for discussion on trust models for the Web.*)

Firstly, we must define the relationship between a class/property term and a vocabulary, and give the notion of *term-level authority*. We view a term as an RDF constant, and a vocabulary as a Web document: from § 2.1, we recall the get mapping from a URI (a Web location) to an RDF graph it may provide by means of a given HTTP lookup, and the redirs mapping for traversing the HTTP redirects given for a URI.

Definition 13 (Authoritative sources for terms). *Letting* $\mathsf{B}(G)$ *denote the set of blank-nodes appearing in the graph* G*, we denote a mapping from a source URI to the set of terms it speaks* authoritatively *for as follows:*[27]

$$\begin{aligned}\mathsf{auth}: \ & \mathsf{S} \to 2^{\mathsf{C}} \\ & s \mapsto \{c \in \mathsf{U} \mid \mathsf{redirs}(c) = s\} \cup \mathsf{B}(\mathsf{get}(s))\end{aligned}$$

Thus, a Web source is authoritative for URIs which dereference to it and the blank nodes it contains; for example, the FOAF vocabulary is authoritative for terms in its namespace since it follows best-practices and makes its class/property URIs dereference

[25] `http://lsdis.cs.uga.edu/~oldham/ontology/wsag/wsag.owl`; retr. early 2010, offline 2011/01/13

[26] `http://www.eiao.net/rdf/1.0`; retr. 2011/01/13

[27] Even predating Linked Data, dereferencable vocabulary terms were encouraged; cf. `http://www.w3.org/TR/2006/WD-swbp-vocab-pub-20060314/`; retr. 2011/01/13

to an RDF/XML document defining the terms. Note that we consider all documents to be non-authoritative for all literals.

To negate the effects of non-authoritative terminological axioms on reasoning over Web data, we add an extra condition to the T-grounding of a rule (see Definition 2): in particular, we only require amendment to rules where both $\mathsf{TBody}(R) \neq \emptyset$ and $\mathsf{ABody}(R) \neq \emptyset$.

Definition 14 (Authoritative T-ground rule instance). *Let* $\mathsf{TAVars}(R) \subset \mathsf{V}$ *denote the set of variables appearing in both* $\mathsf{TBody}(R)$ *and* $\mathsf{ABody}(R)$*, let* G *denote a graph, and let* s *denote the source of that graph. Now, we define the set of authoritative T-ground rule instances for a program* P *in the graph* G *as:*

$\widehat{\mathsf{Ground}}^T(P, G, s) :=$
$\left\{\mathsf{Ground}^T_\theta(\{R\}, G) \mid R \in P^{T\emptyset} \cup P^{\emptyset A} \vee \left(R \in P^{TA} \wedge \exists v \in \mathsf{TAVars}(R) \text{ s.t. } \theta(v) \in \mathsf{auth}(s)\right)\right\}$
where Ground^T_θ *is the T-grounding (as per Definition 2) using the the most general unifier* θ*, and where we recall the* $P^{TA}, P^{T\emptyset}, P^{\emptyset A}$ *conventions from Definition 3.*

The additional condition for authoritativeness states that if $\mathsf{ABody}(R) \neq \emptyset$ and $\mathsf{TBody}(R) \neq \emptyset$, then the unifier θ must substitute at least one variable appearing in both $\mathsf{ABody}(R)$ and $\mathsf{TBody}(R)$ for an authoritative term (wrt. source s)—i.e., source s must speak authoritatively for a term that necessarily appears in each instance of $\mathsf{ABody}(R)$, and cannot create rule instances which could apply over arbitrary assertional data not mentioning any of its terms. We now formalise this notion:

Theorem 4 (Authoritative reasoning guarantee). *Let* Const *denote a function which returns the Herbrand universe of a set of rules (including facts): i.e., a function which returns the set of RDF constants appearing in a program* P *or a graph* G*. Next, let* G' *be any graph, let* s' *be the source of graph* G' *such that* $\mathsf{get}(s') = G'$*, and let* P *be any (T-split) program and* G *be any graph such that*

$$\mathsf{Const}(P \cup G) \cap \mathsf{auth}(s') = \emptyset\,;$$

i.e., neither P *nor* G *contain any terms for which* s' *speaks authoritatively. Finally, let* P' *be the set of partially evaluated rules derived from* G *with respect to* P*, where:*

$$P' := \{R \in \widehat{\mathsf{Ground}}^T(P, G', s') | \mathsf{Body}(R) \neq \emptyset\}$$

Now, it holds that $\mathsf{lm}(P \cup G) = \mathsf{lm}(P \cup P' \cup G)$.

Corollary 2. *Given the same assumption(s) as Theorem* **??***, it also holds that* $\mathsf{lm}^T(P \cup G) = \mathsf{lm}^T(P \cup P' \cup G)$.

Proofs available in [34].

Example 8. Take the T-split rule R_{EX} as before:

$$\texttt{(?x, a, ?c2)} \leftarrow \underline{\texttt{(?c1, rdfs:subClassOf, ?c2)}}, \texttt{(?x, a, ?c1)}$$

and let G_{EX} be the graph from source s:

$$G_{EX} \coloneqq \{\, (\texttt{foaf:Person}, \texttt{rdfs:subClassOf}, \texttt{foaf:Agent}), \\ (\texttt{foaf:Agent}, \texttt{rdfs:subClassOf}, \texttt{dc:Agent}) \,\}$$

Here, $\mathsf{TAVars}(R_{EX}) = \{\texttt{?c1}\}$. Now, for each substitution θ, there must exist $v \in \mathsf{TAVars}(R_{EX})$ such that s speaks authoritatively for $\theta(v)$. In this case, s must speak authoritatively for the `?c1` substitution `foaf:Person` for the rule:

$$(\texttt{?x}, \texttt{a}, \texttt{foaf:Agent}) \leftarrow (\texttt{?x}, \texttt{a}, \texttt{foaf:Person})$$

to be an authoritatively T-ground rule instance, and speak authoritatively for the `?c1` substitution `foaf:Agent` for:

$$(\texttt{?x}, \texttt{a}, \texttt{dc:Agent}) \leftarrow (\texttt{?x}, \texttt{a}, \texttt{foaf:Agent})$$

to be authoritative. In other words, for these T-ground rules to be authoritative, G_{EX} must be served by the document referenced by the FOAF terms—i.e., the FOAF vocabulary. Note that this authoritatively ground rule contains the term `foaf:Agent` in the body, and thus can only generate inferences over graphs containing this term (for which s is authoritative). ◊

For reference, we highlight variables in $\mathsf{TAVars}(R)$ with boldface in the rule tables of Appendix A (only applies to rules with A-atoms *and* T-atoms in the body).

It is worth noting that for rules where $\mathsf{ABody}(R)$ and $\mathsf{TBody}(R)$ are both non-empty, authoritative instantiation of the rule will only consider unifiers for $\mathsf{TBody}(R)$ which come from one source: however, in practice for OWL 2 RL/RDF this is not so restrictive: although $\mathsf{TBody}(R)$ may contain multiple atoms, in such rules $\mathsf{TBody}(R)$ usually refers to an atomic axiom which requires multiple triples to represent—indeed, the OWL 2 Structural Specification [58] enforces usage of blank-nodes and cardinalities on such constructs to ensure that the constituent triples of the multi-triple axiom appear in one source. To take an example, for the T-atoms:

$$(\texttt{?x}, \texttt{owl:hasValue}, \texttt{?y}) \\ (\texttt{?x}, \texttt{owl:onProperty}, \texttt{?p})$$

we would expect `?x` to be ground by a blank-node skolem and thus expect the instance to come from one graph. Although it should be noted that such restrictions do not carry over for OWL 2 Full—which is applicable for arbitrary RDF graphs—it still seems reasonable for us to restrict those OWL 2 Full terminological axioms which require multiple triples to express to be given entirely within one Web document (here, perhaps even making our reasoning more robust).

Note finally that terminological inferences—produced by rules with only T-atoms—are never considered authoritative. Thus, by applying authoritative reasoning, we do not T-ground rules from such facts. For OWL 2 RL/RDF, this only has a "minor" effect on the least model computation since OWL 2 RL/RDF (intentionally) contains redundant rules [24], which allow for deriving the same inferences on a purely assertional level. Along these lines, in Appendix A, Table 20, we list all of the T-atom only rules; assuming that the inferences given by each rule are not considered terminological, we show how the omissions are covered by the recursive application of other assertional rules. We note that we may miss some inferences possible through inference of

rdfs:subClassOf relations between owl:someValuesFrom restriction classes, and also between owl:allValuesFrom restriction classes, since we do not support the respective assertional rules cls-svf1 and cls-avf.

Distributed Reasoning As previously mentioned, Proposition 2 lends itself to a straightforward distribution strategy for applying our A-linear OWL 2 RL/RDF subset. We briefly discuss our distribution strategy, where we use one master machine to compute and coordinate "global knowledge" and use several slave machines to perform tasks in parallel over the bulk of the corpus. We assume that all machines are in a shared-nothing configuration [72]; we also assume that the corpus is evenly split over the slave machines in preparation for reasoning (in our setting, this is the direct result of our distributed crawler), and that the slave machines have roughly even specifications. For more information about our distribution architecture, we refer the interested reader to [34, § 3.6].

As a first step for the distributed reasoning, we extract the T-Box data from each machine, use the master machine to execute the terminological program and create the residual assertional program, and then distribute this assertional program (the proper rules) to each slave machine and let it apply the program independently (and in parallel) over its local segment of the corpus. This process is summarised as follows:

1. **parallel:** identify and separate out the T-Box from the main corpus in parallel on the slave machines;
2. **local:** the master machine then
   (a) gathers and merges the T-Box segments from the slave machines;
   (b) generates axiomatic triples from the meta-program and applies T-atom only rules over the T-Box;
   (c) authoritatively grounds the T-atoms in rules with one A-atom, thus generating the A-linear assertional program;
   (d) optimises the assertional program by merging rules and building a linked rule index;
3. **parallel:** send the assertional linked rule index to all slave machines and reason over the main corpus in parallel on each machine.

The results of the above three-step operation are: (a) axiomatic triples and terminological inferences resident on the master machine; and (b) assertional inferences split over the slave machines. Note further that the output of this process may contain (both local and global) duplicates.

2.6 Linked Data Reasoning Evaluation

We now give evaluation of applying our subset of OWL 2 RL/RDF over the 1.12b quads (947m unique triples) of Linked Data crawled in the previous section. Note that we also require information about redirects encountered in the crawl to reconstruct the redirs function required for authoritative reasoning (see Definition 13) and that we output a flat file of G-Zipped triples. All of our evaluation is based on nine machines

(1 master/8 slaves) connected by Gigabit ethernet[28], each with uniform specifications; viz.: 2.2GHz Opteron x86-64, 4GB main memory, 160GB SATA hard-disks, running Java 1.6.0_12 on Debian 5.0.4. Please note that much of the evaluation presented in this tutorial assumes that the slave machines have roughly equal specifications in order to ensure that tasks finish in roughly the same time, assuming even data distribution.

Survey of Terminology In [34], we presented an analysis of the use of RDFS and OWL in the terminology given by our corpus, in particular with respect to OWL 2 RL/RDF rules. By extension, we provided insights into which RDFS and OWL constructs feature prominently in Linked Data vocabularies. We refer [34] for the details, but in summary we found that our A-linear rules support 99.3% of the total T-ground rules generated from the terminology in the Linked Data corpus, and *authoritative* reasoning with respect to these rules supports 81.7% of the total; excluding one document from the `ontologydesignpatterns.org` domain which publishes 61,887 non-authoritative axioms, the latter percentage increases to 95.1%. Our authoritative A-linear rules fully support (with respect to OWL 2 RL/RDF rules) 90.6% of the documents containing unique terminology, and partially support 99% of these documents. The summation of the ranks of documents fully supported by our A-linear rules was 77% of the total, and the analagous percentage for documents supported by authoritative reasoning over these rules was 70.3% of the total; we found that the top-ranked documents favour RDFS/OWL 1 axioms which are expressible as a single RDF triple (as opposed to, e.g., class descriptions requiring use of lists, or restrictions), and that the highest ranked document serving non-authoritative axioms was FOAF (#7), which makes an `owl:equivalentClass` assertion between `foaf:Agent` and `dct:Agent`, and an `owl:equivalentProperty` assertion between `foaf:maker` and `dct:creator` (in effect, our authoritative reasoning algorithm would treat axioms these as uni-directions sub-class/-property mappings from FOAF to DC).

Authoritative Reasoning In [34], we also compared the effects of authoritative vs. non-authoritative reasoning for our corpus. We refer [34] for the details, but in summary we found that for the instance data of the top five most popular classes and properties, non-authoritative inference sizes are on average 55.46× larger than the authoritative equivalent. Much of this is attributable to noise in and around core RDF(S)/OWL terms, in particular `rdf:type`, `owl:Thing` and `rdfs:Resource`;[29] without these core terms, non-authoritative inferencing creates 12.74× more inferences than the authoritative equivalent.

We present a selected example for the most popular class in our data: `foaf:Person`. Excluding the top-level concepts `rdfs:Resource` and `owl:Thing`, and the inferences possible therefrom, each `rdf:type` triple with `foaf:Person` as value

[28] We observe, e.g., a max FTP transfer rate of 38MB/sec between machines.

[29] We note that much of the noise is attributable to 107 terms from the `opencalais.com` domain; cf. `http://d.opencalais.com/1/type/em/r/PersonAttributes.rdf` (retr. 2011/01/22) and `http://groups.google.com/group/pedantic-web/browse_thread/thread/5e5bd42a9226a419` (retr. 2011/01/22).

Class	(Raw) Count
Authoritative	
foaf:Agent	8,165,989
wgs84:SpatialThing	64,411
contact:Person	1,704
dct:Agent	35
contact:SocialEntity	1
Non-Authoritative (additional)	
po:Person	852
wn:Person	1
aifb:Kategorie-3AAIFB	0
b2r2008:Controlled_vocabularies	0
foaf:Friend_of_a_friend	0
frbr:Person	0
frbr:ResponsibleEntity	0
pres:Person	0
po:Category	0
sc:Agent_Generic	0
sc:Person	0
wn:Agent-3	0

Table 2. Breakdown of non-authoritative and authoritative inferences for `foaf:Person`, with number of appearances as a value for `rdf:type` in the raw data

leads to (at least) five authoritative inferences and twenty-six *additional* non-authoritative inferences (all class memberships). Of the latter twenty-six, fourteen are anonymous classes. Table 2 enumerates the five authoritatively-inferred class memberships and the remaining twelve non-authoritatively inferred *named* class memberships; also given are the occurrences of the class as a value for `rdf:type` in the raw data. Although we cannot claim that all of the additional classes inferred non-authoritatively are *noise*—although classes such as `b2r2008:Controlled_vocabularies` appear to be—we can see that they are infrequently used and arguably *obscure*. Although some of the inferences we omit may of course be serendipitous—e.g., perhaps `po:Person`—again we currently cannot distinguish such cases from noise or blatant spam; for reasons of **robustness** and **terseness**, we conservatively omit such inferences.

Single-machine Reasoning We first applied authoritative reasoning on one machine: reasoning over the dataset described inferred 1.58 billion raw triples, which were filtered to 1.14 billion triples removing non-RDF generalised triples and tautological statements (see § 2.2)—post-processing revealed that 962 million (∼61%) were unique and had not been asserted (roughly a 1:1 *inferred*:*asserted* ratio). The first step—extracting 1.1 million T-Box triples from the dataset—took 8.2 h.

Subsequently, Table 3 gives the results for reasoning on one machine for each approach outlined in § 2.4. T-Box level processing—e.g., applying terminological rules,

partially evaluation, rule indexing, etc.—took roughly the same time (∼9 min) for each approach. During the partial evaluation of the meta-program, 301 thousand assertional rules were created with 2.23 million links; these were subsequently merged down to 216 thousand (71.8%) with 1.15 million (51.6%) links. After saturation, each rule has an average of 6 atoms in the head and all links are successfully removed; however, the saturation causes the same problems with extra duplicate triples as before, and so the fastest approach is PIM, which takes ∼15% of the time for the baseline N algorithm. Note that with 301 thousand assertional rules and without indexing, applying all rules to all statements—roughly 750 trillion rule applications—would take approximately 19 years. In Figure **??**, we also show the linear performance of the fastest approach: PIM (we would expect all methods to be similarly linear).

	T-Box (min)	**A-Box (hr)**
N	8.9	118.4
NI	8.9	121.3
P	8.9	171609[a]
PI	8.9	22.1
PIM	8.9	17.7
PIMS	8.9	19.5

Table 3. Performance for reasoning over 1.1 billion statements on one machine for all approaches

[a] Estimated as a linear product from one day of reasoning.

Fig. 1. Detailed throughput performance for application of assertional program using the fastest approach: PIM

Distributed Reasoning We also apply reasoning over 1, 2, 4 and 8 slave machines using the distribution strategy outlined in § 2.5; Table 4 gives the performance. Note that the most expensive aspects of the reasoning process—extracting the T-Box from the dataset and executing the assertional program—can be executed in parallel by the slave machines without coordination. The only communication required between the machines is during the aggregation of the T-Box and the subsequent partial evaluation and creation of the shared assertional-rule index: this takes ∼10 min, and becomes the lower bound for time taken for distributed evaluation with arbitrary machine count.

In summary, taking our best performance, we apply reasoning over 1.12 billion Linked Data triples in 3.35 h using 9 machines (1 master/8 slaves), deriving 1.58 billion inferred triples, of which 962 million are novel and unique.

2.7 Related Work

Herein, we discuss related works specifically in the field of scalable and distributed reasoning as well as works in the area of robust Web reasoning.

Machines	Extract T-Box	Build T-Box	Reason A-Box	Total
1	492	8.9	1062	1565
2	240	10.2	465	719
4	131	10.4	239	383
8	67	9.8	121	201

Table 4. Distributed reasoning in *minutes* using PIM for 1, 2, 4 & 8 slave machines

Scalable/Distributed Reasoning From the perspective of scalable RDF(S)/OWL reasoning, one of the earliest engines to demonstrate reasoning over datasets in the order of a billion triples was the commercial system BigOWLIM [11], which is based on a scalable and custom-built database management system over which a rule-based materialisation layer is implemented, supporting fragments such as RDFS and pD*, and more recently OWL 2 RL/RDF. Most recent results claim to be able to load 12 billion statements of the LUBM synthetic benchmark, and 20.5 billion statements statements inferrable by pD* rules on a machine with 2x Xeon 5430 (2.5GHz, quad-core), and 64GB (FB-DDR2) of RAM.[30] We note that this system has been employed for relatively high-profile applications, including use as the content management system for a live BBC World Cup site.[31] BigOWLIM features distribution, but only as a replication strategy for fault-tolerance and supporting higher query load.

A number of scalable and distributed reasoners adopt a similar approach to SAOR.

Weaver and Hendler [86] discuss a similar approach for distributed materialisation with respect to RDFS—they also describe a separation of terminological (what they call ontological) data from assertional data. Thereafter, they identify that all RDFS rules have only one assertional atom and, like us, use this as the basis for a scalable distribution strategy: they flood the ontological data and split the assertional data over their machines. They demonstrate the completeness of their approach—arriving to a similar conclusion to us—but by inspection of the RDFS fragment. Inferencing is done over an in-memory RDF store. They evaluate their approach over a LUBM-generated synthetic corpus of 345.5 million triples using a maximum of 128 machines (each with two dual-core 2.6 GHz AMD Opteron processors and 16 GB memory); with this setup, reasoning in memory takes just under 5 minutes, producing 650 million triples.

Similarly following our earlier work on SAOR, Urbani et al. [82] use MapReduce [17] for distributed RDFS materialisation over 850m Linked Data triples. They also consider a separation of terminological (what they call schema) data from assertional data as a core optimisation of their approach, and—likewise with [86]—identify that RDFS rules only contain one assertional atom. As a pre-processing step, they sort their data by subject to reduce duplication of inferences. Based on inspection of the rules, they also identify an ordering (stratification) of RDFS rules which (again assuming standard usage of the RDFS meta-vocabulary) allows for completeness of results without full recursion—unlike us, they do reasoning on a per-rule basis as opposed to

[30] http://www.ontotext.com/owlim/benchmarking/lubm.html; retr. 2011/01/22

[31] http://www.readwriteweb.com/archives/bbc_world_cup_website_semantic_technology.php; retr. 2012/01/22

our per-triple basis. Unlike us, they also use a 8-byte dictionary encoding of terms. Using 32 machines (each with 4 cores and 4 GB of memory) they infer 30 billion triples from 865 million triples in less than one hour; however, they do not materialise or *decode* the output—a potentially expensive process. Note that they do not include any notion of authority (although they mention that in future, they may include such analysis): they attempted to apply pD* on 35 million Web triples and stopped after creating 3.8 billion inferences in 12 h, lending strength to our arguments for authoritative reasoning.

In more recent work, (approximately) the same authors [81] revisit the topic of materialisation with respect to pD*. They again use a separation of terminological data from assertional data, but since pD* contains rules with multiple assertional atoms, they define bespoke MapReduce procedures to handle each such rule, some of which are similar in principle to those presented in [35] (and later on) such as canonicalisation of terms related by `owl:sameAs`. They demonstrate their methods over three datasets; (i) 1.51 billion triples of UniProt data, generating 2.03 billion inferences in 6.1 h using 32 machines; (ii) 0.9 billion triples of LDSR data, generating 0.94 billion inferences in 3.52 h using 32 machines; (iii) 102.5 billion triples of LUBM, generating 47.6 billion inferences in 45.7 h using 64 machines. The latter experiment is two orders of magnitude above our current experiments, and features rules which require A-Box joins; however, the authors do not look at open Web data, stating that:

> *"*[...] *reasoning over arbitrary triples retrieved from the Web would result in useless and unrealistic derivations."*
>
> **—[81]**

They do, however, mention the possibility of including our authoritative reasoning algorithm in their approach, in order to prevent such adverse affects.

In very recent work, [49] have presented an (Oracle) RDBMS-based OWL 2 RL/RDF materialisation approach. They again use some similar optimisations to the scalable reasoning literature, including parallelisation, canonicalisation of `owl:sameAs` inferences, and also partial evaluation of rules based on highly selective patterns—from discussion in the paper, these selective patterns seem to correlate with the terminological patterns of the rule. They also discuss many low-level engineering optimisations and Oracle tweaks to boost performance. Unlike the approaches mentioned thus far, [49] tackle the issue of updates, proposing variants of semi-naïve evaluation to avoid rederivations. The authors evaluate their work for a number of different datasets and hardware configurations; the largest scale experiment they present consists of applying OWL 2 RL/RDF materialisation over 13 billion triples of LUBM using 8 nodes (Intel Xeon 2.53 GHz CPU, 72GB memory each) in just under 2 hours.

Web Reasoning As previously mentioned, [82] discuss reasoning over 850m Linked Data triples—however, they only do so over RDFS and do not consider any issues relating to provenance.

[47] apply reasoning over 0.9 billion Linked Data triples using the aforementioned BigOWLIM reasoner; however, this dataset is manually selected as a merge of a number

of smaller, known datasets as opposed to an arbitrary corpus—they do not consider any general notions of provenance or Web tolerance. (Again, [81] also apply reasoning over the LDSR dataset.)

Related to the idea of authoritative reasoning is the notion of "conservative extensions" described in the Description Logics literature (see, e.g., [21, 54, 43]). However, the notion of a "conservative extension" was defined with a slightly different objective in mind: according to the notion of deductively conservative extensions, a dataset G_a is only considered malicious towards G_b if it causes additional inferences with respect to the intersection of the *signature*—loosely, the set of classes and properties defined in the dataset's namespace—of the original G_b with the newly inferred statements. Thus, for example, defining `ex:moniker` as a *super*-property of `foaf:name` outside of the FOAF spec would be "disallowed" by our authoritative reasoning: however, this would still be a conservative extension since no new inferences using FOAF terms can be created.

Work presented by [14] use a notion of an *authoritative description* which aligns very much with our notion of authority. They use their notion of authority to do reasoning over class hierarchies, but only include custom support of `rdfs:subClassOf` and `owl:equivalentClass`, as opposed to our general framework for authoritative reasoning over arbitrary T-split rules.

A viable alternative approach—which looks more generally at provenance for Web reasoning—is that of "quarantined reasoning", described by [18] and employed by Sindice [63]. The core intuition is to consider applying reasoning on a per-document basis, taking each Web document and its recursive (implicit and explicit) imports and applying reasoning over the union of these documents. The reasoned corpus is then generated as the merge of these per-document closures. Their evaluation was performed in parallel using three machines (quad-core 2.33GHz CPU with 8GB memory each); they reported loading, on average, 40 documents per second.

2.8 Critical Discussion and Future Directions

Herein, we have demonstrated that materialisation with respect to a carefully selected—but still inclusive—subset of OWL 2 RL/RDF rules is currently feasible over large corpora (in the order of a billion triples) of arbitrary RDF data collected from the Web; in order to avoid creating a massive bulk of inferences and to protect popular vocabularies from third-party interference, we include analyses of the source of terminological data into our reasoning, conservatively ignoring third-party contributions and only considering first-party definitions and alignments. Referring back to our motivating `foaf:page` example in the introduction, we can now get the same answers for the simple query if posed over the union of the input and inferred data as for the extended query posed over only the input data.

We do however identify some shortcomings of our approach. Firstly, the scalability of our approach is predicated on the assumption that the terminological fragment of the corpus remain relatively small and simple—as we have seen in § 2.6, this holds true for our current Linked Data corpus. The further from this assumption we get, the closer we get to quadratic (and possibly cubic) materialisation on a terminological level, and a high τ "multiplier" for the assertional program. Thus, the future feasibility of our

approach for the Web (in its current form) depends on the assumption that assertional data dwarves terminological data. We note that almost all highly-scalable approaches in the literature currently rely on a similar premise to some extent, especially for partial-evaluation and distribution strategies.

Secondly, we adopt a very conservative authoritative approach to reasoning which may miss some interesting inferences given by independently published mappings: although we still allow one vocabulary to map its local terms to those of an external vocabulary, we thus depend on each vocabulary to provide all useful mappings in the dereferenced document. In future work, it would be worthwhile to investigate identifying "trusted" third-party mappings in the wild, perhaps based on links-analysis or observed adoption.

Thirdly, thus far we have not considered rules with more than one A-atom—rules which could, of course, lead to useful inferences for our query-answering use-case. Many such rules—for example supporting property-chains, transitivity or equality—can naïvely lead to quadratic inferencing with respect to many reasonable corpora of assertional data. As previously discussed, a backward-chaining or hybrid approach may often make more sense in cases where materialisation produces too many inferences; in fact, we discuss such an approach for equality reasoning in [34]. Note however that not all multiple A-atom rules can produce quadratic inferencing with respect to assertional data: some rules (such as cls-int1, cls-svf1) are what we call *A-guarded*, whereby (loosely) the head of the rule contains only one variable not ground by partial evaluation with respect to the terminology, and thus we posit that such rules also abide by our maximum least-model size for A-linear programs (these are highlighted in Table 18). Despite this, such rules would not fit neatly into our distribution framework (would not be conveniently partitionable), where assertional data must then be coordinated between machines to ensure correct computation of joins (such as in [81]); similarly, some variable portion of assertional data must also be indexed to compute these joins.

Finally, despite our authoritative analysis, reasoning may still introduce significant noise and produce unwanted or unintended consequences; in particular, publishers of assertional data are sometimes unaware of the precise semantics of the vocabulary terms they use. An interesting avenue to explore would be non-standard reasoning approaches (e.g., using statistical models or inductive reasoning) as an alternative or complement to the standard approaches presented herein. (*Please see [73] in these proceedings for discussion on combining probabalistic and logical reasoning for Web data; see [10] in these proceedings for an introduction to scalable non-standard reasoning for the Semantic Web; also, see [28] in these proceedings for discussion on building models for the Web of Data*)

Along similar lines, in [34], we looked at a use-case for annotated reasoning whereby we rank triples in the input data (based on a PageRank analysis of the sources of data) and propagate these ranks to inferences through the annotated reasoning framework. Thereafter, we perform a granular repair of inconsistencies, with the core approach being to removing the weakest triple causing the underlying inconsistency. We refer the interested reader to [34] for more detail.

3 Scalable Approximative OWL 2 DL Reasoning

Ontologies have been so phenomenally successful, as a machine-understandable compilation of human knowledge, that OWL2 (the second version of OWL) is recently standardised by W3C. As more and more large ontologies become available [64], there is a pressing need for efficient and robust reasoning services. Such reasoning services will help us gain insight of the semantic relations among vocabularis of ontologies and facilitate further processings such as the materialisation that we introduced in § 2.1.

Expressive Description Logics (DLs) [5] have high worst case computational complexity. For example, TBox (terminological box) reasoning in the DL $\mathcal{SROIQ}$ [39], the adjacent logic of OWL2-DL, is N2EXPTIME-complete [45]. Mainstream reasoners for expressive DLs provide reasoning services, such as classification (computing subsumption relations among all the named concepts), based on tableau [40] and hypertableau [59] algorithms. Such *model constructing* algorithms classify an ontology, in general, by iterating all necessary pairs of concepts, and trying to construct a model of the ontology that violates the subsumption relation between them [46]. On the other hand, light-weight DLs can have very efficient reasoning algorithms. For example, TBox reasoning in $\mathcal{EL}^{++}$ [3], the logic underpinning of an OWL2 tractable profile OWL2-EL, is PTIME-complete. However, they only provide limited expressive power.

This brings a new challenge: can users use OWL2-DL to build their ontologies and still enjoy the efficient reasoning as in tractable profiles? For example, the Foundational Model of Anatomy ontology (FMA) , which is built in $\mathcal{ALCOIF}$, beyond any tractable DLs, can hardly be classified by any mainstream DL reasoners [60]. Given the current efforts of ontology construction, it might not take long before many other FMA-like (or even larger and more complicated) ontologies appear and go beyond the capability of existing DL reasoners.

Approximation [74, 25, 33, 85, 65] has been identified as a potential way to reduce the complexity of ontology reasoning. However, many of these approximation approaches still rely on the reasoners of the more expressive DLs. For example, [25] replaces certain parts of a concept expression with $\top$ or $\bot$ to obtain a simpler expression that can be classified more easily with a tableau reasoner. [65] requires the use of reasoners of the more expressive DLs to pre-compute the entailments to achieve efficient online performance. Furthermore, most of the above approaches are on ABox reasoning and query answering. To the best of our knowledge, the only approach on TBox reasoning is [25], which presents an overview of approximation approaches (including language weakening, knowledge compilation and approximate deduction), as well as investigating and reporting negative results of the approximate deduction approach – a problematic side effect of using their approximate deduction approach is that the collapsing of concept expressions leading to many unnecessary approximation steps.

In this section, we propose to combine the idea of language weakening and approximate deduction [25] into soundness preserving approximation for ontology TBox reasoning.

1. After an informative discussion of the technical challenges (§ 3.1), we propose a syntactic language weakening approach (§ 3.3, § 3.4 and § 3.5) to approximating an arbitrary $\mathcal{SROIQ}$ TBox with a corresponding $\mathcal{EL}^{++}$ TBox and additional data

structures maintaining the complementary information and cardinality information. It is shown that the proposed approximation is in linear time (Lemma 1, 2 and 3).
2. We present soundness-guaranteed approximate deduction rules to classify the approximated TBox (§ 3.4 and § 3.5). In contrast to the twisted trade-off between tractability and expressiveness, our approach compromises the completeness of reasoning to yield large portion of logical consequences in polynomial time while imposing no restrictions on expressivity of the language used in source ontologies and preserve correctness of results (§ 3.6).
3. We present our implementation and preliminary evaluations (§ 3.7). Evaluation against a set of real world ontologies [64] suggested that, a naive implementation of our approach can (i) outperform existing OWL2-DL reasoners such as Pellet and FaCT++, and (ii) provide rather complete results with high recall (over 95% for $\mathcal{EL}^{++}_{\mathcal{C}}$ and over 99% for $\mathcal{EL}^{++}_{\mathcal{CQ}}$, where $\mathcal{EL}^{++}_{\mathcal{C}}$ and $\mathcal{EL}^{++}_{\mathcal{CQ}}$ are two more and more fine-grained approximation).

Proofs of all propositions, lemmas and theorems can be found in tech report available at `http://www.box.net/shared/nm913g22ie`.

3.1 Technical Motivations

In order to motivate our investigation on syntactic approximation of $\mathcal{SROIQ}$ ontologies to $\mathcal{EL}^{++}$ ontologies, this section first briefly introduces $\mathcal{SROIQ}$ and $\mathcal{EL}^{++}$ and then illustrates the technical challenges in their TBox reasoning and approximation.

In $\mathcal{SROIQ}$, concept C, D can be inductively composed with the following constructs:

$$\top \mid \bot \mid A \mid C \sqcap D \mid \exists R.C \mid \{a\} \mid \neg C \mid \geq nR.C \mid \exists R.Self$$

where $\top$ is the top concept, $\bot$ the bottom concept, A atomic concept, n an integer number, a an individual, $\exists R.Self$ the self-restriction and R a role that can be either an atomic role r or the inverse of another role (R^-). Conventionally, $C \sqcup D, \forall R.C$ and $\leq nR.C$ are used to abbreviate $\neg(\neg C \sqcap \neg D), \neg\exists R.\neg C$ and $\neg \geq (n+1)R.C$, respectively. $\{a_1, a_2, \ldots, a_n\}$ can be regarded as abbreviation of $\{a_1\} \sqcup \{a_2\} \sqcup \cdots \sqcup \{a_n\}$. Without loss of generality, in what follows, we assume all the concepts to be in their negation normal forms (NNF)[32] and use ~C to denote the NNF of $\neg C$. We also call $\top, \bot, A, \{a\}$ *basic concepts* because they are not composed by other concepts or roles. Given a TBox T, we use CN_T (RN_T) to denote the set of basic concepts (atomic roles) in T. The $\mathcal{EL}$ family is dedicated for large TBox reasoning and has been widely applied in some largest ontologies, e.g. SNOMED [71]. $\mathcal{EL}^{++}$ supports

$$\top \mid \bot \mid A \mid C \sqcap D \mid \exists r.C \mid \{a\}.$$

Both $\mathcal{SROIQ}$ and $\mathcal{EL}^{++}$ support concept inclusions (CIs, e.g. $C \sqsubseteq D$) and role inclusions (RIs, e.g. $r \sqsubseteq s$, $r_1 \circ \cdots \circ r_n \sqsubseteq s$). $\mathcal{SROIQ}$ supports also other axioms

[32] An $\mathcal{SROIQ}$ concept is in NNF *iff* negation is applied only to atomic concepts, nominals or Self-restriction. NNF of a given concept can be computed in linear time[38].

such asymmetric of roles. If $C \sqsubseteq D$ and $D \sqsubseteq C$, we write $C \equiv D$. If C is non-atomic, $C \sqsubseteq D$ is a general concept inclusion (GCI). For more details about syntax and semantics of DLs, we refer the readers to [69] in these proceedings and [5].

A TBox is a set of concept and role axioms. TBox reasoning services include concept subsumption checking, concept satisfiability checking (to check if a given concept is instantiatable) and classification (to compute the concept hierarchy). For example, given the following TBox $\mathcal{T}_1$ (in $\mathcal{ALC}$), we can infer $Koala \sqsubseteq Herbivore$.

Example 9. An example TBox $\mathcal{T}_1$.

- $\alpha_1 : Koala \sqsubseteq \forall eat.(\exists partof.Eucalypt)$
- $\alpha_2 : Eucalypt \sqsubseteq Plant$
- $\alpha_3 : Plant \sqcup \exists partof.Plant \sqsubseteq VegeFood$
- $\alpha_4 : \forall eat.VegeFood \sqsubseteq Herbivore$

The tableau algorithm [40] constructs a tableau (as a witness of a model of the TBox $\mathcal{T}_1$) as a graph in which each node x represents an individual and is labelled with a set of concepts it must satisfy, each edge $\langle x, y \rangle$ represents a pair of individuals satisfying a role that labels the edge. Subsumption checking $C \sqsubseteq D$ can be reduced to unsatisfiability checking $C \sqcap \neg D \sqsubseteq \bot$. To test this, a tableau is initialised with a single node labelled with $C \sqcap \neg D$, and is then expanded by repeatedly applying the completion rules [40].

One of the major difficulties for tableau algorithms is the high degree of non-determinism introduced by GCIs. For each GCI $C \sqsubseteq D$ in the ontology, the algorithm generates a meta-constraint $\neg C \sqcup D$ for each node of the tableau. The algorithm first extends a node with $\neg C$. If it finds a clash, it backtracks and extends the node with D. If there are n GCIs, this expands to 2^n combinations for each node of the tableau. This significantly enlarges the search space.

Some techniques have been developed to deal with GCIs. *Absorption* [79] can reduce, e.g. a GCI $A \sqcap C \sqsubseteq D$, where A is a named concept, into non-GCI $A \sqsubseteq \neg C \sqcup D$; however, it is only applicable for GCIs whose LHS is a conjunction with a named concept as conjunct or whose RHS is a negated named concept or a disjunction with a negated named concept as disjunct. *(Extended) Role Absorption* [78, Sec.4.1] can absorb GCIs of form $\exists r.C \sqsubseteq D$ ($C \sqsubseteq \forall r.D$) into domain (range) constrains. For example α_3 can be decomposed into $\exists partof.Plant \sqsubseteq VegeFood$ and thus absorbed as $Domain(partof, VegeFood \sqcup \neg \exists partof.Plant)$. But its applicability is still limited and it still contains a disjunction in the domain. *Binary Absorption* [42] tries to rewrite GCIs into form $A_1 \sqcap A_2 \sqsubseteq C$ where A_1 and A_2 are named concepts. To sum up, the above absorptions can only be applied to a limited patterns of GCIs; e.g., α_4 can not be dealt with by any absorption optimisation.

Reasoning with $\mathcal{EL}^{++}$ is more efficient. [3] presents a set of completion rules (Table 5) [33] to compute, given a normalised $\mathcal{EL}^{++}$ TBox T, for each $A \in CN_T$, a subsumer set $S(A) \subseteq CN_T \cup \{\bot\}$ in which for each $B \in S(A)$, $T \models A \sqsubseteq B$, and for each $r \in RN_T$, a relation set $R(r) \subseteq CN_T \times CN_T$ in which for each $(A, B) \in R(r)$, $T \models A \sqsubseteq \exists r.B$.

[33] in **R6** $X \rightsquigarrow_R A$ iff there exists $C_1, \ldots, C_k \in CN_T$ s.t. $C_1 = X$ or $C_1 = \{b\}$, $(C_j, C_{j+1}) \in R(r_j)$ for some $r_j \in RN_T (1 \leq j \leq k)$ and $C_k = A$

R1	If $A \in S(X)$, $A \sqsubseteq B \in \mathcal{T}$ and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
R2	If $A_1, A_2, \ldots, A_n \in S(X)$, $A_1 \sqcap A_2 \sqcap \cdots \sqcap A_n \sqsubseteq B \in \mathcal{T}$ and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
R3	If $A \in S(X)$, $A \sqsubseteq \exists r.B \in \mathcal{T}$ and $(X, B) \notin R(r)$ then $R(r) := R(r) \cup \{(X, B)\}$
R4	If $(X, A) \in R(r)$ $A' \in S(A)$, $\exists r.A' \sqsubseteq B \in \mathcal{T}$ and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
R5	If $(X, A) \in R(r)$, $\bot \in S(A)$ and $\bot \notin S(X)$ then $S(X) := S(X) \cup \{\bot\}$
R6	If $\{a\} \in S(X) \cap S(A)$, $X \rightsquigarrow_R A$ and $S(A) \not\subseteq S(X)$ then $S(X) := S(X) \cup S(A)$
R7	If $(X, A) \in R(r)$, $r \sqsubseteq s \in \mathcal{T}$ and $(X, A) \notin R(s)$ then $R(s) := R(s) \cup \{(X, A)\}$
R8	If $(X, A) \in R(r_1)$, $(A, B) \in R(r_2)$, $r_1 \circ r_2 \sqsubseteq r_3 \in \mathcal{T}$, and $(X, B) \notin R(r_3)$ then $R(r_3) := R(r_3) \cup \{(X, B)\}$

Table 5. $\mathcal{EL}^{++}$ completion rules (no datatypes)

Reasoning with rules **R1-R8** is tractable. However, these rules can not handle $\mathcal{T}_1$ because the ontology is in a language beyond the $\mathcal{EL}^{++}$.

Groot et al. [25] attempt to speed up concept unsatisfiability checking via approximation. Given a concept C, it constructs a sequence of $C_i^{\top}$ such that $C \sqsubseteq \cdots \sqsubseteq C_1^{\top} \sqsubseteq C_0^{\top}$, and a sequence of $C_i^{\bot}$ such that $C_0^{\bot} \sqsubseteq C_1^{\bot} \sqsubseteq \ldots C$ by replacing all existential restrictions ($\exists R.D$) after i universal quantifiers ($\forall$) inside C with $\top$ and $\bot$ respectively. Then C is unsatisfiable (satisfiable) if some $C_i^{\top}$ ($C_i^{\bot}$) is unsatisfiable (satisfiable). In case $C_i^{\top}$ ($C_i^{\bot}$) is usually simpler than C, its (un)satisfiability checking should also be easier. For example, a concept $C \equiv \neg Herbivore \sqcap \forall eat.(VegeFood \sqcap \exists partof.Plant)$ can be approximated to $C_1^{\top} \equiv \neg Herbivore \sqcap \forall eat.(VegeFood \sqcap \top) \equiv \neg Herbivore \sqcap \forall eat.VegeFood$, which is unsatisfiable in $\mathcal{T}_1$. Thus C is unsatisfiable. However, this approach has several limitations when applied to TBox reasoning:

1. It only approximates the tested concept, but not the ontology, thus the unsatisfiability checking still requires reasoners for the original language of the ontology. In other words, it does not reduce the complexity of reasoning.
2. Similar to the Tableau algorithms, to classify an ontology, one has to reduce concept subsumption $C \sqsubseteq D$ to unsatisfiability of $C \sqcap \neg D$ for each necessary pair of C, D.
3. When the test concept subsumption contains no existential restriction, such as $Koala \sqsubseteq Herbivore$, this approach can not help. Hence, it does not help for classification (subsumption checking among named concepts).

Due to the above reasons, this approximation technology is not suitable for TBox Reasoning, especially computing the atomic concept hierarchies.

To sum up, tableau algorithms have difficulties to handle complex structured axioms; tractable DL algorithms can not support more expressive languages; while traditional approximation approach lacks usability in TBox reasoning. In what follows, we present our approach which is motivated and inspired by these works, and show that it overcomes these difficulties with evaluations.

3.2 Approach Overview and Preliminary

Different from Groot et al.'s approximation approach, we approximate both the ontology and the tested concept (if needed) by replacing concept sub-expressions (role expressions) that are not in the target DL, e.g. $\mathcal{EL}^{++}$, with atomic concepts (atomic roles) and rewrite axioms accordingly (§ 3.3). Then, additional data structures and completion rules (§ 3.4 and § 3.5) are used to maintain and restore some semantic relations among basic concepts, respectively. We show that all these approachs are tractable and soundness-guaranteed (§ 3.6).

In approximation, we only consider concepts corresponding to the particular TBox in question. We use the notion *term* to refer to these "interesting" concept expressions. More precisely, a term is:

1. a concept expression on the LHS or RHS of any CI, or
2. the singleton of any individual in the ontology, or
3. the syntactic sub-expression of a term, or
4. the complement of a term.

In order to represent all these terms and role expressions that will be used in $\mathcal{EL}^{++}$ reasoning, we first assign names to them.

Definition 15. ***(Name Assignment)*** *Given S a set of concept expressions, E a set of role expressions, a* name assignment *fn is a function as for each $C \in S$ ($R \in E$), $fn(C) = C$ ($fn(R) = R$) if C is a basic concept (R is atomic), otherwise $fn(C)$ ($fn(R)$) is a fresh name.*

As an example, name assignments of some terms in Example 9 are illustrated in Table 6.

From § 3.1 we can see that there is an expressivity gap between $\mathcal{SROIQ}$ and $\mathcal{EL}^{++}$, especially in concept constructs. In the rest of this section, we present 3 stages of approximation to (partially) bridge this gap.

3.3 $\mathcal{EL}^{++}$ Approximation

A naive $\mathcal{EL}^{++}$ approximation is to approximate an arbitrary TBox into an $\mathcal{EL}^{++}$ TBox.

Definition 16. ***($\mathcal{EL}^{++}$ Transformation)*** *Given a TBox $\mathcal{T}$ and a name assignment fn, its $\mathcal{EL}^{++}$* transformation *$A_{fn,\mathcal{EL}^{++}}(\mathcal{T})$ is a set of axiom T constructed as follows:*

1. *T is initialised as $\emptyset$.*
2. *for each $C \sqsubseteq D$ ($C \equiv D$) in $\mathcal{T}$, $T = T \cup \{fn(C) \sqsubseteq fn(D)\}$ ($T = T \cup \{fn(C) \equiv fn(D)\}$).*

Term	Name
$\forall eat.\exists partof.Eucalypt$	C_1
$\exists eat.\forall partof.\neg Eucalypt$	nC_1
$\forall partof.\neg Eucalypt$	C_2
$\exists partof.Eucalypt$	nC_2
$Plant \sqcup \exists partof.Plant$	C_3
$\neg Plant \sqcap \forall partof.\neg Plant$	nC_3
$\forall partof.\neg Plant$	C_4
$\exists partof.Plant$	nC_4
$\forall eat.VegeFood$	C_5
$\exists eat.\neg VegeFood$	nC_5
$\neg Plant$	$nPlant$
$\neg VegeFood$	$nVegeFood$

Table 6. Name Assignment

3. *for each $\mathcal{EL}^{++}$ role axiom $\beta \in \mathcal{T}$, add $\beta_{[R/fn(R)]}$ into T.*
4. *for each term C in $\mathcal{T}$,*
   (a) if C is the form $C_1 \sqcap \cdots \sqcap C_n$, then $T = T \cup \{fn(C) \equiv fn(C_1) \sqcap \cdots \sqcap fn(C_n)\}$,
   (b) if C is the form $\exists R.D$, then $T = T \cup \{fn(C) \equiv \exists fn(R).fn(D)\}$,
   (c) otherwise $T = T \cup \{fn(C) \sqsubseteq \top\}$.

In the above definition, Step 2 rewrites all the concept axioms; Step 3 rewrites all the $\mathcal{EL}^{++}$ role axioms; Step 4 rewrites all the $\mathcal{EL}^{++}$ terms. We call this procedure an $\mathcal{EL}^{++}$ *approximation*.

Lemma 1. *For a TBox $\mathcal{T}$ and a name assignment fn, let $A_{fn,\mathcal{EL}^{++}}(\mathcal{T}) = T$. Then T is an $\mathcal{EL}^{++}$ TBox and $|T| \leq n_{\mathcal{T}} + |\mathcal{T}|$ where $n_{\mathcal{T}}$ is the number of terms in $\mathcal{T}$ and $|T|$ ($|\mathcal{T}|$) is the number of axioms in T ($\mathcal{T}$) .*

According to Table 6, we can transform the TBox $\mathcal{T}_1$ into T_{Koala} as follows:

Example 10. T_{Koala} contains axioms generated by Step 2 and 4, the most important ones include:

- $\alpha_1 \rightarrow Koala \sqsubseteq C_1$, $nC_1 \equiv \exists eat.C_2$, $nC_2 \equiv \exists partof.Eucalypt$;
- α_2 is preserved;
- $\alpha_3 \rightarrow C_3 \sqsubseteq VegeFood$, $nC_3 \equiv nPlant \sqcap C_4$, $nC_4 \equiv \exists partof.Plant$;
- $\alpha_4 \rightarrow C_5 \sqsubseteq Herbivore$, $nC_5 \equiv \exists eat.nVegeFood$.

3.4 Complement-enriched $\mathcal{EL}^{++}_{\mathcal{C}}$ Approximation

In Example 10, reasoning can be performed directly with the completion rules **R1-R8** presented in Table 5. However, $T_{Koala} \not\models Koala \sqsubseteq Herbivore$ because the relations between a term and its complement, e.g. C_1 and nC_1, can not be directly represented in $\mathcal{EL}^{++}$. To solve this problem, we maintain such relations in a separate *complement table (CT)*, and apply additional completion rules in reasoning.

Approximate Complement We first extend the naive $\mathcal{EL}^{++}$ approximation with a complement table (CT).

Definition 17. *($\mathcal{EL}^{++}_{\mathcal{C}}$ **Transformation**) Given a TBox $\mathcal{T}$ and a name assignment fn, its complement-enriched $\mathcal{EL}^{++}_{\mathcal{C}}$* transformation $A_{fn,\mathcal{EL}^{++}_{\mathcal{C}}}(\mathcal{T})$ *is a pair (T, CT) constructed as follows:*

1. $T = A_{fn,\mathcal{EL}^{++}}(\mathcal{T})$ *(Ref. Def. 16).*
2. CT *is initialised as* $\emptyset$.
3. *for each term C in $\mathcal{T}$, $CT = CT \cup \{(fn(C), fn(\sim C))\}$.*

We call this procedure an $\mathcal{EL}^{++}_{\mathcal{C}}$ *approximation*. The following proposition shows the structure of the approximation results:

Proposition 4. *($\mathcal{EL}^{++}_{\mathcal{C}}$ **Approximation**) For a TBox $\mathcal{T}$, let $A_{fn,\mathcal{EL}^{++}_{\mathcal{C}}}(\mathcal{T}) = (T, CT)$, we have:*

1. *T is an $\mathcal{EL}^{++}$ TBox*
2. *for each $A \in CN_T$, there exists $(A, B) \in CT$*
3. *if $(A, B) \in CT$ then $A, B \in CN_T$ and $(B, A) \in CT$*

This indicates that, by Def.17, a TBox can be syntactically transformed into an $\mathcal{EL}^{++}$ TBox with a table maintaining complementary relations for all names in the $\mathcal{EL}^{++}$ TBox.

Example 11. The $\mathcal{EL}^{++}_{\mathcal{C}}$ approximation of $\mathcal{T}_1$ in Example 9 is (T_{Koala}, CT_{Koala}), where T_{Koala} is the same as in Example 10, and CT_{Koala} contains pairs such as $(C_1, nC_1), (C_2, nC_2), (C_3, nC_3), (C_4, nC_4), (C_5, nC_5), (Plant, nPlant), (VegeFood, nVegeFood)$, etc.

Lemma 2. *For any TBox $\mathcal{T}$ and (T, CT) its $\mathcal{EL}^{++}_{\mathcal{C}}$ approximation, if $\mathcal{T}$ contains $n_{\mathcal{T}}$ terms, then $|T| \leq n_{\mathcal{T}} + |\mathcal{T}|$ and $|CT| = n_{\mathcal{T}}$, where $|T|(|\mathcal{T}|)$ is the number of axioms in $T(\mathcal{T})$ and $|CT|$ is the number of pairs in CT.*

Completion Rules for Complement Given an $\mathcal{EL}^{++}_{\mathcal{C}}$ transformation (T, CT), we normalise axioms of form $C \sqsubseteq D_1 \sqcap \cdots \sqcap D_n$ into $C \sqsubseteq D_1, \ldots, C \sqsubseteq D_n$, and recursively normalise role chain $r_1 \circ \cdots \circ r_n \sqsubseteq s$ with $n > 2$ into $r_1 \circ \cdots \circ r_{n-1} \sqsubseteq u$ and $u \sqsubseteq s$. This procedure can be done in linear time. In the following, we assume T to be always normalised. For convenience, we use a *complement function* $fc : CN_T \mapsto CN_T$ as: for each $A \in CN_T$, $fc(A) = B$ such that $(A, B) \in CT$.

To utilise the complementary relations in CT, we propose additional completion rules (Table 7) to $\mathcal{EL}^{++}$.

R9 realises axiom $A \sqcap \sim A \sqsubseteq \bot$. **R10** asserts the reverse subsumption between concepts to supplement the absence of negation, i.e. $A \sqsubseteq B \rightarrow \sim A \sqsubseteq \sim B$. **R11** builds up the relations between conjuncts of a conjunction, *e.g.* $A \sqcap B \sqsubseteq \bot$ implies $A \sqsubseteq \sim B$.

Now we can infer $Koala \sqsubseteq Herbivore$ (Example 11) as follows:

– $\alpha_2 \rightarrow nC_2 \sqsubseteq nC_4 \rightarrow_{R10} C_4 \sqsubseteq C_2 \rightarrow nC_3 \sqsubseteq C_2$

R9	If $A, B \in S(X)$, $A = fc(B)$ and $\bot \notin S(X)$ then $S(X) := S(X) \cup \{\bot\}$
R10	If $A \in S(B)$ and $fc(B) \notin S(fc(A))$ then $S(fc(A)) := S(fc(A)) \cup \{fc(B)\}$
R11	If $A_1 \sqcap \cdots \sqcap A_i \sqcap \cdots \sqcap A_n \sqsubseteq \bot$, $A_1, \ldots, A_{i-1}$, $A_{i+1}, \ldots, A_n \in S(X)$ and $fc(A_i) \notin S(X)$ then $S(X) := S(X) \cup \{fc(A_i)\}$

Table 7. Complement completion rules

- $C_3 \sqsubseteq VegeFood \rightarrow_{R10} nVegeFood \sqsubseteq nC_3$
- $nVegeFood \sqsubseteq nC_3$, $nC_3 \sqsubseteq C_2 \rightarrow nVegeFood \sqsubseteq C_2 \rightarrow nC_5 \sqsubseteq nC_1 \rightarrow_{R10} C_1 \sqsubseteq C_5 \rightarrow Koala \sqsubseteq Herbivore$

where the inferences with $\rightarrow_{R10}$ are enabled by **R10**.

3.5 Cardinality-enriched $\mathcal{EL}^{++}_{\mathcal{CQ}}$ Approximation

In Def.17 we presented an extension of the naive approximation which approximates non-$\mathcal{EL}^+$ concept expressions, particularly concepts constructed by $\neg$, $\sqcup$ and $\forall$, by the definition of their complements. With the completion rules in Tab.7, more entablements can be computed.

It is a natural question to ask, is that possible to approximate even more non-$\mathcal{EL}^{++}$ construct, e.g. cardinality, into $\mathcal{EL}^{++}$? In this subsection, we further extend the $\mathcal{EL}^{++}_{\mathcal{C}}$ transformation to yield more complete reasoning results for ontology containing cardinalities.

Approximating Cardinality In $\mathcal{EL}^{++}_{\mathcal{C}}$ approximation, a concept constructed by $\geq$ or $\leq$ can only be represented as a fresh name. In this way, subsumption $X \sqsubseteq \bot$ can not be entailed in $\mathcal{T}_4$ in the following Example 12.

Example 12. $\mathcal{T}_4 = \{X \sqsubseteq \geq 4r.A, X \sqsubseteq \leq 2s.B, A \sqsubseteq B, r \sqsubseteq s\}$.
$X \sqsubseteq \bot$ should be entailed.

This subsumption requires to maintain the relations between the filler concepts (e.g. A and B), the role (r) and the cardinality values (e.g 4 and 2). We maintain such relations in a *(*cardinality table) (QT) whose elements are tuples (A, r, n), where A is a basic concept denoting a filler name, r is the atomic role denoting the role name and n is the cardinality value.

Definition 18. ***(Cardinality-enriched $\mathcal{EL}^{++}_{\mathcal{CQ}}$ Transformation)*** *Given a TBox $\mathcal{T}$, a name assignment fn, let $A_{fn,\mathcal{EL}^{++}_{\mathcal{C}}}(\mathcal{T}) = (T', CT')$, its cardinality-enriched $\mathcal{EL}^{++}_{\mathcal{CQ}}$* transformation *$A_{fn,\mathcal{EL}^{++}_{\mathcal{CQ}}}(\mathcal{T})$ is a tuple (T, CT, QT) constructed as follows:*

1. *T is initialised as T'.*
2. *$CT = CT'$.*

3. *QT is initialised as $\emptyset$.*
4. *for each term C that is the form $\geq nR.D$ in $\mathcal{T}$,*
   (a) *if $n = 0$, $T = T \cup \{\top \sqsubseteq fn(C)\}$*
   (b) *if $n = 1$, $T = T \cup \{fn(C) \equiv \exists fn(R).fn(D)\}$*
   (c) *otherwise, $T = T \cup \{fn(C) \equiv fn(D)^{fn(R),n}\}$, and $QT = QT \cup \{(fn(C), fn(R), n)\}$.*
5. *for each pair of names A and r, if there exist $(A, r, i_1), (A, r, i_2), \ldots, (A, r, i_n) \in QT$ with $i_1 < i_2 < \cdots < i_n$, $T = T \cup \{A^{r,i_n} \sqsubseteq A^{r,i_{n-1}}, \ldots, A^{r,i_2} \sqsubseteq A^{r,i_1}, A^{r,i_1} \sqsubseteq \exists r.A\}$*

In step 4, $fn(D)^{fn(R),n}$ is a fresh name. For example, $nVegeFood^{eat,3}$ for $\geq 3eat.\neg VegeFood$. Obviously, this is unique for a given tuple of D, R and n. Similarly, $\leq nR.D$ will be approximated via the approximation of its complement $\geq (n+1)R.D$. In step 5, for each pair of name assignment A, r in T, a subsumption chain is added into T because $\geq i_n r.A \sqsubseteq \cdots \sqsubseteq \geq i_2 r.A \sqsubseteq \geq i_1 r.A \sqsubseteq \exists r.A$.

We call this procedure an $\mathcal{EL}^{++}_{\mathcal{CQ}}$ *approximation*. The following proposition shows the structure of the results:

Proposition 5. ***($\mathcal{EL}^{++}_{\mathcal{CQ}}$ Approximation)***
For a TBox $\mathcal{T}$, a name assignment fn, let $A_{fn,\mathcal{EL}^{++}_{\mathcal{CQ}}}(\mathcal{T}) = (T, CT, QT)$, we have T an $\mathcal{EL}^{++}$ TBox.

This indicates that, by Def.18 a TBox can be syntactically transformed into a tuple of an $\mathcal{EL}^{++}$ TBox, a complement table and a cardinality table.

Now, in Example 12, $\mathcal{T}_4$ can be approximated into $T_4 \supseteq \{X, \sqsubseteq Y_1, Y_1 \equiv A^{r,4}, X \sqsubseteq Y_2, nY_2 \equiv B^{s,3}, A \sqsubseteq B, r \sqsubseteq s\}$ with $fn(\geq 4r.A) = Y_1$, $fn(\leq 2s.B) = Y_2$ and $fn(\geq 3s.B) = nY_2$, $CT_4 \supseteq \{(Y_1, nY_1), (Y_2, nY_2)\}$, $QT_4 \supseteq \{(A, r, 4), (B, s, 3)\}$.

Lemma 3. *For any TBox $\mathcal{T}$, let (T, CT, QT) its $\mathcal{EL}^{++}_{\mathcal{CQ}}$ transformation, if $\mathcal{T}$ contains $n_{\mathcal{T}}$ terms, then $|CN_T| \leq 2 \times n_{\mathcal{T}}$, $|T| \leq 3 \times n_{\mathcal{T}} + |\mathcal{T}|$, $|CT| = n_{\mathcal{T}}$ and $|QT| \leq n_{\mathcal{T}}$, where CN_T is the number of basic concepts in T, $|T|(|\mathcal{T}|)$ the number of axioms in $T(\mathcal{T})$, $|CT|$ the number of pairs in CT and $|QT|$ the number of tuples in QT.*

Completion rules We further extend Tab.7 with Table 8.

R12, in which $r \sqsubseteq_* s$ if $r = s$ or $r \sqsubseteq s \in T$, realises inference $A \sqsubseteq B, R \sqsubseteq S, i \geq j \rightarrow \geq iR.A \sqsubseteq \geq jS.B$. **R13** is the extension of **R4** and **R14-16** are extensions of **R8**. Now we can entail $X \sqsubseteq \bot$ in Example 12 as follows:

1. $A \sqsubseteq B, r \sqsubseteq s \rightarrow_{R12} A^{r,4} \sqsubseteq B^{s,3}$,
2. $A^{r,4} \sqsubseteq B^{s,3}, X \sqsubseteq Y_1, Y_1 \equiv A^{r,4}, nY_2 \equiv B^{s,3} \rightarrow X \sqsubseteq nY_2$
3. $X \sqsubseteq nY_2, X \sqsubseteq Y_2, (Y_2, nY_2) \in CT \rightarrow_{R9} X \sqsubseteq \bot$

3.6 Reasoning Properties

In this subsection, we analyze the reasoning complexity of our approximation and reasoning approach.

R12	If $B \in S(A), (A, r, i), (B, s, j) \in QT, r \sqsubseteq_* s$, $i \geq j$ and $B^j \notin S(A^i)$ then $S(A^{r,i}) := S(A^{r,i}) \cup \{B^{s,j}\}$
R13	If $A^{r,i} \in S(X), A' \in S(A), \exists r.A' \sqsubseteq B \in T$ and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
R14	If $A^{r_1,i} \in S(X), (A, B) \in R(r_2), r_1 \circ r_2 \sqsubseteq r_3 \in T$, and $(X, B) \notin R(r_3)$ then $R(r_3) := R(r_3) \cup \{(X, B)\}$
R15	If $(X, A) \in R(r_1), B^{r_2,i} \in S(A), r_1 \circ r_2 \sqsubseteq r_3 \in \mathcal{T}$, and $(X, B) \notin R(r_3)$ then $R(r_3) := R(r_3) \cup \{(X, B)\}$
R16	If $A^{r_1,i} \in S(X), B^{r_2,j} \in S(A), r_1 \circ r_2 \sqsubseteq r_3 \in \mathcal{T}$, and $(X, B) \notin R(r_3)$ then $R(r_3) := R(r_3) \cup \{(X, B)\}$

Table 8. Cardinality completion rule

Theorem 5. ***(Complexity)*** *For any $\mathcal{EL}^{++}_{\mathcal{CQ}}$ transformation (T, CT, QT) (T normalised), TBox reasoning by completion rules **R1-R16** will terminate in polynomial time w.r.t. $|CN_T|$+$|RN_T|$.*

Similarly, reasoning on the $\mathcal{EL}^{++}$ and $\mathcal{EL}^{++}_{\mathcal{C}}$ approximations are also tractable. Note that, from lemma 1, 2 and 3, the approximation is always linear. To sum up, the approximation-reasoning approach is tractable.

With the approximation and corresponding rules, we can compute concept subsumption in an $\mathcal{SROIQ}$ TBox. The quality of the approximate reasoning is described by the following theorem:

Theorem 6. ***(Concept Subsumption Checking)*** *Given a TBox $\mathcal{T}$, its vocabulary $V_{\mathcal{T}}$ and $A_{fn,\mathcal{EL}^{++}_{\mathcal{CQ}}} = (T, CT, QT)$, for any two concepts C and D constructed from $V_{\mathcal{T}}$, if $A_{fn,\mathcal{EL}^{++}_{\mathcal{CQ}}}(\{C \sqsubseteq \top, D \sqsubseteq \top\}) = (T', CT', QT')$, then $\mathcal{T} \models C \sqsubseteq D$ if $fn(D) \in S(fn(C))$ can be computed by rules **R1-R16** on $(T \cup T', CT \cup CT', QT \cup QT')$.*

The theorem indicates that our $\mathcal{EL}^{++}_{\mathcal{CQ}}$ approximate reasoning approach is soundness-preserving. This conclusion holds similarly on $\mathcal{EL}^{++}$ and $\mathcal{EL}^{++}_{\mathcal{C}}$ approximate reasoning.

Particularly, when C, D are terms in $\mathcal{T}$, $\mathcal{T} \models C \sqsubseteq D$ if $fn(D) \in S(fn(C))$ can be derived from (T, CT, QT).

As in classical reasoning, unsatisfiability checking of a concept C can be reduced to entailment checking of $C \sqsubseteq \bot$; ontology inconsistency checking can be reduced to entailment checking of $\top \sqsubseteq \bot$ or $\{a\} \sqsubseteq \bot$. By applying ABox internaliation, ABox reasoning can be reduced to TBox reasoning, e.g. $a : A$ if $A \in S(\{a\})$ can be computed. For more optimised approach on ABox reasoning with syntactic approximation, we refer readers to [68].

More extension patterns can be exploit to improve the completeness of the approximate reasoning while keep it tractable. Our framework is flexible and extendible.

Our extra completion rules process each axiom and term in T individually. This helps keeping the reasoning tractable but some information can be lost:

Example 13. $\mathcal{T}_5 = \{A \sqcap \neg B \sqsubseteq C, A \sqcap B \sqsubseteq C, D \sqsubseteq \exists r.\top, \exists r.C \sqsubseteq E, \exists r.\neg A \sqsubseteq E\}$

Obviously, we have $\mathcal{T}_5 \models A \sqsubseteq C$ and thus $D \sqsubseteq E$.

We approximate $\mathcal{T}_5$ into $(\{X_1 \equiv A \sqcap nB, X_1 \sqsubseteq C, X_2 \equiv A \sqcap B, X_2 \sqsubseteq C, X_3 \equiv \exists r.nA, X_3 \sqsubseteq E, \dots\}, \{(B, nB), \dots\})$. Our approach will reach $B \sqcap A \sqsubseteq C$ and $nB \sqcap A \sqsubseteq C$. Because B and nB are not subsumers of A thus we can't infer $C \in S(A)$. Even if we can compute it, in order to further infer $D \sqsubseteq F$. A new axiom $\exists r.(C \sqcup \neg A) \equiv \exists r.C \sqcup \exists r.\neg A$ has to be added into $\mathcal{T}$ and approximated for incremental reasoning.

Although we do not guarantee completeness, we will see in next section it has high recall for many test ontologies.

3.7 Evaluation

We implemented 3 versions of our approach, namely the $\mathcal{EL}^{++}$, $\mathcal{EL}^{++}_{\mathcal{C}}$, $\mathcal{EL}^{++}_{\mathcal{CQ}}$ approximation and reasoning systems, in the TrOWL tractable reasoning infrastructure [34]. To evaluate their performance in practice, we compared with mainstream reasoners Pellet 2.0.0, FaCT++ 1.3.0.1 and HermiT 1.1. All experiments were conducted in an environment of Windows XP SP3 with 2.66 GHz CPU and 1G RAM allocated to JVM 1.6.0.07.

Following [60], we examined the most difficult ontologies in the HermiT benchmark [64]. To focus on TBox reasoning, we removed the ABox axioms with care[35] from these ontologies. Most of the remaining TBoxes can be classified easily by all the reasoners and completely by our $\mathcal{EL}^{++}_{\mathcal{C}}$ system. We evaluate the hard ones, results shown in Table 9 and 10. We mainly conducted the evaluations on $\mathcal{EL}^{++}_{\mathcal{C}}$ system. To show the effects of complement-enriched approximate reasoning, we present also the $\mathcal{EL}^{++}$ recall. For those TBoxes that the $\mathcal{EL}^{++}_{\mathcal{C}}$ provides incomplete classification, we further classified them with the $\mathcal{EL}^{++}_{\mathcal{CQ}}$ system.

Ontology $\mathcal{O}$	$\lvert\mathcal{O}\rvert$	FaCT++	HermiT	Pellet
Biological Process	32289	3.656	5.343	10.063
Cellular Component	47348	5.872	8.077	16.966
GO	32289	18.563	6.047	16.39
Cyc	11727	25.531	16.853	142.889
FMA Constitutional	123564	e/o	e/o	e/o
Tambis Full	606	0.375	1.063	1.343
Wine	454	0.578	0.875	1.359
DLP	1216	0.219	61.948	98.024

Table 9. Results of main stream reasoners

[34] `http://trowl.eu`

[35] ABox axioms involving individuals appearing in the TBox were internalised, e.g. $a : C$ into $\{a\} \sqsubseteq C, a \neq b$ into $\{a\} \sqcap \{b\} \sqsubseteq \bot$, etc.. The others are removed.

Ontology	$\mathcal{EL}^{++}$	$\mathcal{EL}^{++}_{\mathcal{C}}$		$\mathcal{EL}^{++}_{\mathcal{CQ}}$	
	recall	time	recall	time	recall
Biological Process	93.1%	1.11	100%	-	-
Cellular Component	91.9%	1.359	100%	-	-
GO	93.1%	4.203	100%	-	-
Cyc	1.2%	1.672	100%	-	-
FMA Constitutional	N/A	10.062	N/A	50.89	N/A
Tambis Full	7.2%	0.11	99.3%	0.203	100%
Wine	95.8%	0.078	96.8%	0.156	99.4%
DLP	100%	0.125	100%	-	-

Table 10. Results of our systems

Each reasoner was given 10 min to classify each ontology. We queried for subsumption relations between named concepts (including *owl:Thing* and *owl:Nothing*) and counted the numbers. Recall of our systems is computed against others to measure the completeness. Thus the time shown in our evaluation includes classification time, subsumption retrieval and counting time. Time unit is second.

Results illustrated in Table 10 show that with extension of the approximation, higher and higher recall can be achieved. $\mathcal{EL}^{++}$ is naive and quite incomplete on some ontologies. $\mathcal{EL}^{++}_{\mathcal{C}}$ approximation can significantly improve the recall on some ontologies (such as Cyc and Tambis Full). With further extension to $\mathcal{EL}^{++}_{\mathcal{C}}$ approximation, all the recalls are over 99% (except FMA). Comparison with results illustrated in Table 9 shows that the efficiency of our systems is in general better than all other reasoners. Even the slowest $\mathcal{EL}^{++}_{\mathcal{CQ}}$ system is faster than all main stream reasoners. Also, our systems are the only reasoners that can return result on the FMA ontology.

We were also interested in the scalability of our approach. Based on Table 9 we chose 3 easiest ontologies and enlarged them by duplicating all the concept names (but keep the role names). Duplications were distinguished by a subscript. Consequently, all the concept axioms were duplicated. We classified these ontologies using our $\mathcal{EL}^{++}_{\mathcal{C}}$ system, which has a nice balance between efficiency and completeness (Ref. Table 10). It performed quite stable when the quantity of data increased (Table 11). Due to the interactions between duplications through role axioms, our system even gained some recall on Wine.

Due to the lack of OWL2-DL benchmarks, we turned to ontologies generated from realistic use cases. In [88] an approach of using DL to model relation-based access control (relBAC) has been presented. In this paper, a rather expressive DL $\mathcal{ALCQIBO}$ has been employed to encode various access control schemata. For evaluation purpose, we generated 100 TBoxes containing the following patterns:

1. "User in U are allowed to access (with P) at most n objects in O": $U \sqsubseteq \leq nP.O$
2. "Users in U have access to at least m objects in O with P": $U \sqsubseteq \geq nP.O$
3. "User u is of user type U": $\{u\} \sqsubseteq U$

where U is a type of users, P a permission type, O a object type while u a individual user. Each of these 100 ontologies contains 20 user types, 20 object types, 10 permission

Size	FaCT++	HermiT	Pellet	$\mathcal{EL}^{++}_{\mathcal{C}}$	Recall
Tambis Full					
5×	9.125	37.922	24.25	0.719	99.3%
10×	40.577	292.481	205.192	1.985	99.3%
20×	e/o	t/o	t/o	5.671	N/A
30×	e/o	t/o	t/o	11.624	N/A
Wine					
5×	13.784	56.853	86.662	0.641	97.7%
10×	33.01	t/o	t/o	2.188	97.9%
20×	243.496	t/o	t/o	10.077	98.0%
30×	t/o	t/o	t/o	27.529	N/A
DLP					
5×	t/o	e/o	e/o	3.39	N/A
10×	t/o	e/o	e/o	20.827	N/A
20×	t/o	e/o	e/o	142.305	N/A
30×	t/o	e/o	e/o	450.6	N/A

Table 11. Comparison on duplicated TBox

types, 750 individuals and 20 access control model axioms (axioms of type 1 and 2). Hierarchies among users types (object types) are randomly generated. The numbers in cardinality restrictions are randomly selected. The combinations of user individual, user type class, permission type class and object type class are also random. Obviously, these TBoxes are in DL $\mathcal{ALHOQ}$.

Different from previous evaluations, the TBoxes generated here can be inconsistent. For example, when a particular user belongs to two types U_1 and U_2 with $U_1 \sqsubseteq \leq mP.O_1$ and $U_2 \sqsubseteq \geq nP.O_2$ where $m < n$ and $O_2 \sqsubseteq O_1$, inconsistency occurs. Thus, in this evaluation, we are particularly interested in whether the inconsistency can be detected instead of the number of subsumptions.

We classified these TBoxes using FaCT++, $\mathcal{EL}^{++}_{\mathcal{C}}$ and $\mathcal{EL}^{++}_{\mathcal{CQ}}$ systems. Each reasoner was given 10 minutes. FaCT++ finished 98 of them, failing the other 2. $\mathcal{EL}^{++}_{\mathcal{C}}$ classified all the TBoxes but failed to find any inconsistency, because it does not support cardinality at all. $\mathcal{EL}^{++}_{\mathcal{CQ}}$ classified all the TBoxes efficiently and reported all the inconsistencies correctly. The average and maximal time of FaCT++ and $\mathcal{EL}^{++}_{\mathcal{CQ}}$ and the precisions of the 98 ontologies are illustrated in Table 12. Time unit is second. Notice that in FaCT++, reasoning is immediately terminated when any inconsistency is detected, which means the reasoning time of inconsistent TBox is shorter. While our $\mathcal{EL}^{++}_{\mathcal{CQ}}$ continues to find all the inconsistency. Therefore we separate the results of consistent and inconsistent TBoxes.

The results show that, $\mathcal{EL}^{++}_{\mathcal{CQ}}$ system can classify all the ontologies very efficiently and the presicion is 100%. Also, the average and maximal time is quite stable no matter the ontology is consistent or not. While FaCT++ has difficulty in dealing with consistent TBox containning many cardinality restrictions (Max. time is about 10 seconds and failed on two other TBoxes). For those inconsistent ones, even though FaCT++ terminates earlier, $\mathcal{EL}^{++}_{\mathcal{CQ}}$ system can still outperform it.

Consistency	FaCT++		$\mathcal{EL}^{++}_{\mathcal{CQ}}$		
	Ave.	Max.	Ave.	Max.	presicion
Consistent	1.226	9,984	0.021	0.047	100%
Inconsistent	0.248	2.297	0.022	0.047	100%

Table 12. Comparison on relBAC TBox

3.8 Discussions

Approximate reasoning has been an important topic for ontology (KR) and AI research. On the one hand, expressive Description Logics (such as those underpin the standard Semantic Web ontology languages) have high worst case computational complexity. Hence, approximate reasoning is an attractive way to provide scalable and efficient reasoning services [65]. On the other hand, it has been argued that [25] while logic has always aimed at modelling idealised forms of reasoning under idealised circumstances, this is not what is required under the practical circumstances in knowledge-based systems. Instead, we also need to consider (i) reasoning under time-pressure, (ii) reasoning with other limited resources besides time and (iii) reasoning that is not perfect but instead good enough for given tasks under given circumstances.

In this section, we address a long-lasting open problem; i.e, effective and efficient approximate TBox reasoning. With their negative results, Groot et al. concluded that traditional approximation method by Cadoli and Schaerf [70] is not suited for ontology reasoning, and that new approximate strategy are needed. In this paper, we propose to combine the ideas of language weakening and approximate deduction to provide soundness preserving TBox reasoning for expressive Description Logics. We apply our idea to approximate OWL2-DL ontologies to $\mathcal{EL}^{++}$ ones, preliminary evaluation results showed that our approach performs effectively and efficiently on real world ontologies.

In the approximate deduction step, instead of simplifying a model constructing algorithm (such as tableau algorithm), we enrich the existing $\mathcal{EL}^{++}$ reasoning algorithm with some deterministic completion rules (for complement and cardinality). $\mathcal{EL}^{++}$ retain tractability by imposing strict syntactic restriction. However these restrictions are not always necessary. For example, if we rewrite each axiom $C \sqsubseteq D$ of an $\mathcal{EL}^{++}$ ontology into $\neg D \sqsubseteq \neg C$, the language appears to be $\mathcal{ALC}$, but the complexity does not essentially change. Our approximation can naturally cover these situations. As our evaluation shows, it helps increase the recall.

This piece of work is also related to Horn $\mathcal{SHIQ}$, which has an even more complicated set of syntactic restrictions, which can not be satisfied by our Koala example (more precisely, axioms α_4). In [46], *structure transformation* is applied in a similar manner as our approximation to facilitate reasoning. However *structure transformation* still preserves the syntactic structure of the axioms, while our approximation actually changed the structures and hence ontology such as the Koala example can be classified with a more efficient algorithm.

References

1. D. Allemang and J. A. Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann/Elsevier, 2008.
2. S. Auer, A.-C. N. Ngomo, and J. Lehmann. Introduction to Linked Data. In *Reasoning Web*. Springer, 2011. **(In these proceedings)**.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ Envelope. In *Proceedings IJCAI-05*, 2005.
4. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Application*. Cambridge University Press, 2002.
5. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
6. A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
7. D. Beckett and T. Berners-Lee. Turtle – Terse RDF Triple Language. W3C Team Submission, Jan. 2008. `http://www.w3.org/TeamSubmission/turtle/`.
8. N. Belnap. A Useful Four-Valued Logic. *Modern Uses of Multiple-Valued Logic*, pages 5–37, 1977.
9. T. Berners-Lee. Linked Data. W3C Design Issues, July 2006. From `http://www.w3.org/DesignIssues/LinkedData.html`; retr. 2010/10/27.
10. A. Bernstein. Scalable non-standard reasoning on the Semantic Web. In *Reasoning Web*. Springer, 2011. **(In these proceedings)**.
11. B. Bishop, A. Kiryakov, D. Ognyanoff, I. Peikov, Z. Tashev, and R. Velkov. OWLIM: A family of scalable semantic repositories. *Semantic Web Journal*, 2011. In press; available at `http://www.semantic-web-journal.net/sites/default/files/swj97_0.pdf`.
12. C. Bizer, R. Cyganiak, and T. Heath. How to Publish Linked Data on the Web. `linkeddata.org` Tutorial, July 2008. `http://linkeddata.org/docs/how-to-publish`.
13. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A crystallization point for the Web of Data. *J. Web Sem.*, 7(3):154–165, 2009.
14. 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.
15. S. S. Cosmadakis, H. Gaifman, P. C. Kanellakis, and M. Y. Vardi. Decidable Optimization Problems for Database Logic Programs (Preliminary Report). In *STOC*, pages 477–490, 1988.
16. J. de Bruijn and S. Heymans. Logical foundations of (e)rdf(s): Complexity and reasoning. In *ISWC/ASWC*, pages 86–99, 2007.
17. J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*, pages 137–150, 2004.
18. R. Delbru, A. Polleres, G. Tummarello, and S. Decker. Context Dependent Reasoning for Semantic Documents in Sindice. In *Proc. of 4th SSWS Workshop*, 2008.
19. D. Fensel and F. van Harmelen. Unifying Reasoning and Search to Web Scale. *IEEE Internet Computing*, 11(2):96, 94–95, 2007.
20. R. T. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. J. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, June 1999. `http://www.ietf.org/rfc/rfc2616.txt`.
21. S. Ghilardi, C. Lutz, and F. Wolter. Did i damage my ontology? a case for conservative extensions in description logics. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 187–197, June 2006.

22. B. Glimm. Using SPARQL with RDFS and OWL entailment. In *Reasoning Web*. Springer, 2011. **(In these proceedings)**.
23. C. Golbreich and E. K. Wallace. OWL 2 Web Ontology Language: New Features and Rationale. W3C Recommendation, Oct. 2009. http://www.w3.org/TR/owl2-new-features/.
24. B. C. Grau, B. Motik, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language: Profiles. W3C Recommendation, Oct. 2009. http://www.w3.org/TR/owl2-profiles/.
25. P. Groot, H. Stuckenschmidt, and H. Wache. Approximating Description Logic Classification for Semantic Web Reasoning. In *ESWC-2005*, 2005.
26. 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.
27. Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3):158–182, 2005.
28. C. Gutierrez. Models for the Web of Data. In *Reasoning Web*. Springer, 2011. **(In these proceedings)**.
29. C. Gutierrez. Pascal Hitzler. In *Reasoning Web*. Springer, 2011. **(In these proceedings)**.
30. P. Hayes. RDF Semantics. W3C Recommendation, Feb. 2004. http://www.w3.org/TR/rdf-mt/.
31. M. Hepp. Product Variety, Consumer Preferences, and Web Technology: Can the Web of Data Reduce Price Competition and Increase Customer Satisfaction? In *EC-Web*, page 144, 2009.
32. P. Hitzler and F. van Harmelen. A Reasonable Semantic Web. *Semantic Web Journal – Interoperability, Usability, Applicability*, 1(1), 2010.
33. P. Hitzler and D. Vrandecic. Resolution-Based Approximate Reasoning for OWL DL. In *ISWC-2005*, 2005.
34. A. Hogan. *Exploiting RDFS and OWL for Integrating Heterogeneous, Large-Scale, Linked Data Corpora*. PhD thesis, Digital Enterprise Research Institute, National University of Ireland, Galway, 2011. Available from http://aidanhogan.com/docs/thesis/.
35. A. Hogan, A. Harth, and A. Polleres. Scalable Authoritative OWL Reasoning for the Web. *Int. J. Semantic Web Inf. Syst.*, 5(2), 2009.
36. A. Hogan, A. Harth, J. Umbrich, S. Kinsella, A. Polleres, and S. Decker. Searching and Browsing Linked Data with SWSE: the Semantic Web Search Engine. Technical Report DERI-TR-2010-07-23, Digital Enterprise Research Institute, Galway, 2010. http://www.deri.ie/fileadmin/documents/DERI-TR-2010-07-23.pdf.
37. A. Hogan, J. Z. Pan, A. Polleres, and S. Decker. SAOR: Template Rule Optimisations for Distributed Reasoning over 1 Billion Linked Data Triples. In *International Semantic Web Conference*, 2010.
38. B. Hollunder, W. Nutt, and M. Schmidt-Schau. Subsumption Algorithms for Concept Description Languages. In *ECAI-90*, pages 348–353. Pitman Publishing, 1990.
39. I. Horrocks, O. Kutz, and U. Sattler. The Even More Irresistible SROIQ. In *KR 2006*, 2006.
40. I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8:2000, 2000.
41. Z. Huang and F. van Harmelen. Using Semantic Distances for Reasoning with Inconsistent Ontologies. In *International Semantic Web Conference*, pages 178–194, 2008.
42. E. K. Hudek and G. Weddell. Binary Absorption in Tableaux-Based Reasoning for Description Logics. In *Proc. DL 2006*, 2006.
43. E. Jiménez-Ruiz, B. C. Grau, U. Sattler, T. Schneider, and R. B. Llavori. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *Proceedings of the 21st International Workshop on Description Logics (DL2008)*, May 2008.
44. N. D. Jones, C. K. Gomard, P. Sestoft, L. O. Andersen, and T. Mogensen. *Partial Evaluation and Automatic Program Generation*. Prentice Hall International, 1993.

45. Y. Kazakov. SRIQ and SROIQ are Harder than SHOIQ. In *DL 2008*, 2008.
46. Y. Kazakov. Consequence-Driven Reasoning for Horn SHIQ Ontologies. In *IJCAI 2009*, 2009.
47. A. Kiryakov, D. Ognyanoff, R. Velkov, Z. Tashev, and I. Peikov. LDSR: a Reason-able View to the Web of Linked Data. In *Semantic Web Challenge (ISWC2009)*, 2009.
48. G. Kobilarov, T. Scott, Y. Raimond, S. Oliver, C. Sizemore, M. Smethurst, C. Bizer, and R. Lee. Media Meets Semantic Web - How the BBC Uses DBpedia and Linked Data to Make Connections. In *ESWC*, pages 723–737, 2009.
49. V. Kolovski, Z. Wu, and G. Eadon. Optimizing Enterprise-scale OWL 2 RL Reasoning in a Relational Database System. In *International Semantic Web Conference*, 2010.
50. H. J. Komorowski. Partial Evaluation as a Means for Inferencing Data Structures in an Applicative Language: A Theory and Implementation in the Case of Prolog. In *POPL*, pages 255–267, 1982.
51. D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-Tolerant Semantics for Description Logics. In *RR*, pages 103–117, 2010.
52. J. W. Lloyd. *Foundations of Logic Programming (2nd edition)*. Springer-Verlag, 1987.
53. J. W. Lloyd and J. C. Shepherdson. Partial Evaluation in Logic Programming. *J. Log. Program.*, 11(3&4):217–242, 1991.
54. C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 453–458, January 2007.
55. Y. Ma and P. Hitzler. Paraconsistent Reasoning for OWL 2. In *RR*, pages 197–211, 2009.
56. F. Maier. Extending Paraconsistent $\mathcal{SROIQ}$. In *RR*, pages 118–132, 2010.
57. B. Motik. *Web Ontology Reasoning with Logic Databases*. PhD thesis, AIFB, Karlsruhe, Germany, 2004.
58. B. Motik, P. F. Patel-Schneider, and B. Parsia. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C Recommendation, Oct. 2009. http://www.w3.org/TR/owl2-syntax/.
59. B. Motik, R. Shearer, and I. Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. In *CADE-21*, volume 4603, pages 67–83, 2007.
60. B. Motik, R. Shearer, and I. Horrocks. Hypertableau Reasoning for Description Logics, 2008. Submitted to a journal.
61. S. Muñoz, J. Pérez, and C. Gutiérrez. Minimal Deductive Systems for RDF. In *ESWC*, pages 53–67, 2007.
62. S. Muñoz, J. Pérez, and C. Gutierrez. Simple and Efficient Minimal RDFS. *J. Web Sem.*, 7(3):220–234, 2009.
63. E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: a document-oriented lookup index for open linked data. *IJMSO*, 3(1):37–52, 2008.
64. Oxford-Benchmark. Oxford Benchmark. `http://hermit-reasoner.com/2009/JAIR_benchmarks/`, 2009.
65. J. Z. Pan and E. Thomas. Approximating OWL-DL Ontologies. In *AAAI-2007*, pages 1434–1439, 2007.
66. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, Jan. 2008. `http://www.w3.org/TR/rdf-sparql-query/`.
67. R. Ramakrishnan, D. Srivastava, and S. Sudarshan. Rule Ordering in Bottom-Up Fixpoint Evaluation of Logic Programs. In *Proc. of 16th VLDB*, pages 359–371, 1990.
68. Y. Ren, J. Z. Pan, and Y. Zhao. Towards soundness preserving approximation for abox reasoning of owl2. In *Description Logics Workshop 2010 (DL2010)*, 2010.
69. S. Rudolph. Foundations of Description Logics. In *Reasoning Web*. Springer, 2011. **(In these proceedings)**.

70. M. Schaerf and M. Cadoli. Tractable Reasoning via Approximation. *Artificial Intelligence*, 74:249–310, 1995.
71. K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *JAMIA*, 2000.
72. M. Stonebraker. The Case for Shared Nothing. *IEEE Database Eng. Bull.*, 9(1):4–9, 1986.
73. H. Stuckenschmidt and M. Niepert. Combining Probabilistic and Logical Reasoning for Web Data Processing. In *Reasoning Web*. Springer, 2011. **(In these proceedings)**.
74. H. Stuckenschmidt and F. van Harmelen. Approximating Terminological Queries. In *FQAS '02*, London, UK, 2002.
75. 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.
76. H. J. ter Horst. Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3:79–115, 2005.
77. D. Trček. Trust management methodologies for the Web. In *Reasoning Web*. Springer, 2011. **(In these proceedings)**.
78. D. Tsarkov and I. Horrocks. Efficient Reasoning with Range and Domain Constraints. In *(DL 2004)*, 2004.
79. D. Tsarkov, I. Horrocks, and P. F. Patel-Schneider. Optimizing Terminological Reasoning for Expressive Description Logics. *J. Autom. Reason.*, 39(3):277–316, 2007.
80. J. D. Ullman. *Principles of Database and Knowledge Base Systems*. Computer Science Press, 1989.
81. J. Urbani, S. Kotoulas, J. Maassen, F. van Harmelen, and H. E. Bal. OWL Reasoning with WebPIE: Calculating the Closure of 100 Billion Triples. In *ESWC (1)*, pages 213–227, 2010.
82. J. Urbani, S. Kotoulas, E. Oren, and F. van Harmelen. Scalable Distributed Reasoning Using MapReduce. In *International Semantic Web Conference*, pages 634–649, 2009.
83. R. J. Vanderbei. *Linear Programming: Foundations and Extensions, 3rd ed.* Springer Verlag, 2008.
84. D. Vrandečíc, M. Krötzsch, S. Rudolph, and U. Lösch. Leveraging Non-Lexical Knowledge for the Linked Open Data Web. *Review of April Fool's day Transactions (RAFT)*, 5:18–27, 2010.
85. H. Wache, P. Groot, and H. Stuckenschmidt. Scalable Instance Retrieval for the Semantic Web by Approximation. In *WISE Workshops-05*, 2005.
86. J. Weaver and J. A. Hendler. Parallel Materialization of the Finite RDFS Closure for Hundreds of Millions of Triples. In *International Semantic Web Conference (ISWC2009)*, pages 682–697, 2009.
87. E. Yardeni and E. Y. Shapiro. A Type System for Logic Programs. *J. Log. Program.*, 10(1/2/3&4):125–153, 1991.
88. R. Zhang, A. Artale, F. Giunchiglia, and B. Crispo. Using description logics in relation based access control. In B. C. Grau, I. Horrocks, B. Motik, and U. Sattler, editors, *Description Logics*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.
89. X. Zhang, G. Xiao, and Z. Lin. A Tableau Algorithm for Handling Inconsistency in OWL. In *ESWC*, pages 399–413, 2009.

A OWL 2 RL/RDF rules

Herein, we list the rule tables categorising supported OWL 2 RL/RDF rules [24] according to terminological and assertional arity of atoms in the body. Note that herein we (ab)use Turtle syntax [7] and highlight authoritative variable positions ($\mathsf{TAVars}(R)$—see § 2.5) in bold.

A.1 "A-Linear" OWL 2 RL/RDF Rules

Body(R) $= \emptyset$		
ID	**Head**	**Notes**
prp-ap	*?p* a owl:AnnotationProperty	For each built-in annotation property
cls-thing	owl:Thing a owl:Class .	-
cls-nothing	owl:Nothing a owl:Class .	-
dt-type1	*?dt* a rdfs:Datatype .	For each built-in datatype
~~dt-type2~~[a]	~~*?l* a *?dt* .~~	~~For all *?l* in the value-space of datatype *?dt*~~
~~dt-eq~~[a]	~~$?l_1$ owl:sameAs $?l_2$.~~	~~For all $?l_1$ and $?l_2$ with the same data value~~
~~dt-diff~~[a]	~~$?l_1$ owl:differentFrom $?l_2$.~~	~~For all $?l_1$ and $?l_2$ with different data values~~

Table 13. OWL 2 RL/RDF rules with empty body (axiomatic triples)

[a] These rules mandate (naïvely) infinite materialised inferences, and so we exclude them.

$\mathsf{TBody}(R) \neq \emptyset, \mathsf{ABody}(R) = \emptyset$		
ID	**Body**	**Head**
	terminological	
cls-oo	*?c* owl:oneOf (*?x*$_1$...*?x*$_n$) .	*?x*$_1$...*?x*$_n$ a *?c* .
scm-cls	*?c* a owl:Class .	*?c* rdfs:subClassOf *?c* , owl:Thing ; owl:equivalentClass *?c* . owl:Nothing rdfs:subClassOf *?c* .
scm-sco	*?c*$_1$ rdfs:subClassOf *?c*$_2$. *?c*$_2$ rdfs:subClassOf *?c*$_3$.	*?c*$_1$ rdfs:subClassOf *?c*$_3$.
scm-eqc1	*?c*$_1$ owl:equivalentClass *?c*$_2$.	*?c*$_1$ rdfs:subClassOf *?c*$_2$. *?c*$_2$ rdfs:subClassOf *?c*$_1$.
scm-eqc2	*?c*$_1$ rdfs:subClassOf *?c*$_2$. *?c*$_2$ rdfs:subClassOf *?c*$_1$.	*?c*$_1$ owl:equivalentClass *?c*$_2$.
scm-op	*?p* a owl:ObjectProperty .	*?p* rdfs:subPropertyOf *?p* . *?p* owl:equivalentProperty *?p* .
scm-dp	*?p* a owl:DatatypeProperty .	*?p* rdfs:subPropertyOf *?p* . *?p* owl:equivalentProperty *?p* .
scm-spo	*?p*$_1$ rdfs:subPropertyOf *?p*$_2$. *?p*$_2$ rdfs:subPropertyOf *?p*$_3$.	*?p*$_1$ rdfs:subPropertyOf *?p*$_3$.
scm-eqp1	*?p*$_1$ owl:equivalentProperty *?p*$_2$.	*?p*$_1$ rdfs:subPropertyOf *?p*$_2$. *?p*$_2$ rdfs:subPropertyOf *?p*$_1$.
scm-eqp2	*?p*$_1$ rdfs:subPropertyOf *?p*$_2$. *?p*$_2$ rdfs:subPropertyOf *?p*$_1$.	*?p*$_1$ owl:equivalentProperty *?p*$_2$.
scm-dom1	*?p* rdfs:domain *?c*$_1$. *?c*$_1$ rdfs:subClassOf *?c*$_2$.	*?p* rdfs:domain *?c*$_2$.
scm-dom2	*?p*$_2$ rdfs:domain *?c* . *?p*$_1$ rdfs:subPropertyOf *?p*$_2$.	*?p*$_1$ rdfs:domain *?c* .
scm-rng1	*?p* rdfs:range *?c*$_1$. *?c*$_1$ rdfs:subClassOf *?c*$_2$.	*?p* rdfs:range *?c*$_2$.
scm-rng2	*?p*$_2$ rdfs:range *?c* . *?p*$_1$ rdfs:subPropertyOf *?p*$_2$.	*?p*$_1$ rdfs:range *?c* .
scm-hv	*?c*$_1$ owl:hasValue *?i* ; owl:onProperty *?p*$_1$. *?c*$_2$ owl:hasValue *?i* ; owl:onProperty *?p*$_2$. *?p*$_1$ rdfs:subPropertyOf *?p*$_2$.	*?c*$_1$ rdfs:subClassOf *?c*$_2$.
scm-svf1	*?c*$_1$ owl:someValuesFrom *?y*$_1$; owl:onProperty *?p* . *?c*$_2$ owl:someValuesFrom *?y*$_2$; owl:onProperty *?p* . *?y*$_1$ rdfs:subClassOf *?y*$_2$.	*?c*$_1$ rdfs:subClassOf *?c*$_2$.
scm-svf2	*?c*$_1$ owl:someValuesFrom *?y* ; owl:onProperty *?p*$_1$. *?c*$_2$ owl:someValuesFrom *?y* ; owl:onProperty *?p*$_2$. *?p*$_1$ rdfs:subPropertyOf *?p*$_2$.	*?c*$_1$ rdfs:subClassOf *?c*$_2$.
scm-avf1	*?c*$_1$ owl:allValuesFrom *?y*$_1$; owl:onProperty *?p* . *?c*$_2$ owl:allValuesFrom *?y*$_2$; owl:onProperty *?p* . *?y*$_1$ rdfs:subClassOf *?y*$_2$.	*?c*$_1$ rdfs:subClassOf *?c*$_2$.
scm-avf2	*?c*$_1$ owl:allValuesFrom *?y* ; owl:onProperty *?p*$_1$. *?c*$_2$ owl:allValuesFrom *?y* ; owl:onProperty *?p*$_2$. *?p*$_1$ rdfs:subPropertyOf *?p*$_2$.	*?c*$_1$ rdfs:subClassOf *?c*$_2$.
scm-int	*?c* owl:intersectionOf (*?c*$_1$...*?c*$_n$) .	*?c* rdfs:subClassOf *?c*$_1$...*?c*$_n$.
scm-uni	*?c* owl:unionOf (*?c*$_1$...*?c*$_n$) .	*?c*$_1$...*?c*$_n$ rdfs:subClassOf *?c* .

Table 14. OWL 2 RL/RDF rules containing only T-atoms in the body

$\mathsf{ABody}(R) \neq \emptyset, \mathsf{TBody}(R) = \emptyset$		
ID	**Body**	**Head**
	assertional	
~~eq-ref~~[a]	~~*?s ?p ?o .*~~	~~*?s* owl::sameAs *?s* . *?p* owl::sameAs *?p* . *?o* owl::sameAs *?o* .~~
eq-sym	*?x* owl::sameAs *?y* .	*?y* owl::sameAs *?x* .

Table 15. OWL 2 RL/RDF rules with no T-atoms, but one A-atom in the body

[a] We typically omit this rule which adds unnecessary bulk to the materialised inferences, and could be more easily supported by backward-chaining.

$\mathsf{TBody}(R) \neq \emptyset$ and $\lvert\mathsf{ABody}(R)\rvert = 1$			
ID	**Body**		**Head**
	terminological	*assertional*	
prp-dom	***?p*** rdfs:domain *?c* .	*?x ?p ?y* .	*?x* a *?c* .
prp-rng	***?p*** rdfs:range *?c* .	*?x ?p ?y* .	*?y* a *?c* .
prp-symp	***?p*** a owl:SymmetricProperty .	*?x ?p ?y* .	*?y ?p ?x* .
prp-spo1	$\boldsymbol{?p_1}$ rdfs:subPropertyOf $?p_2$.	*?x* $?p_1$ *?y* .	*?x* $?p_2$ *?y* .
prp-eqp1	$\boldsymbol{?p_1}$ owl:equivalentProperty $?p_2$.	*?x* $?p_1$ *?y* .	*?x* $?p_2$ *?y* .
prp-eqp2	$?p_1$ owl:equivalentProperty $\boldsymbol{?p_2}$.	*?x* $?p_2$ *?y* .	*?x* $?p_1$ *?y* .
prp-inv1	$\boldsymbol{?p_1}$ owl:inverseOf $?p_2$.	*?x* $?p_1$ *?y* .	*?y* $?p_2$ *?x* .
prp-inv2	$?p_1$ owl:inverseOf $\boldsymbol{?p_2}$.	*?x* $?p_2$ *?y* .	*?y* $?p_1$ *?x* .
cls-int2	***?c*** owl:intersectionOf ($?c_1$ ... $?c_n$) .	*?x* a *?c* .	*?x* a $?c_1$, ..., $?c_n$.
cls-uni	*?c* owl:unionOf ($?c_1$...$\boldsymbol{?c_i}$...$?c_n$) .	*?x* a $?c_i$	*?x* a *?c* .
cls-svf2	*?x* owl:someValuesFrom owl:Thing ; owl:onProperty ***?p*** .	*?u ?p ?v* .	*?u* a *?x* .
cls-hv1	***?x*** owl:hasValue *?y* ; owl:onProperty ***?p*** .	*?u* a *?x* .	*?u ?p ?y* .
cls-hv2	*?x* owl:hasValue ***?y*** ; owl:onProperty ***?p*** .	*?u ?p ?y* .	*?u* a *?x*
cax-sco	$\boldsymbol{?c_1}$ rdfs:subClassOf $?c_2$.	*?x* a $?c_1$.	*?x* a $?c_2$.
cax-eqc1	$\boldsymbol{?c_1}$ owl:equivalentClass $?c_2$.	*?x* a $?c_1$.	*?x* a $?c_2$.
cax-eqc2	$?c_1$ owl:equivalentClass $\boldsymbol{?c_2}$.	*?x* a $?c_2$.	*?x* a $?c_1$.

Table 16. OWL 2 RL/RDF rules containing some T-atoms and precisely one A-atom in the body

A.2 Unsupported OWL 2 RL/RDF Rules

$\lvert\mathsf{ABody}(R)\rvert > 1, \mathsf{TBody}(R) = \emptyset$		
ID	**Body**	**Head**
	assertional	
eq-trans	*?x* owl:sameAs *?y* . *?y* owl:sameAs *?z* .	*?x* owl:sameAs *?z* .
eq-rep-s	*?s* owl:sameAs *?s′* . *?s ?p ?o* .	*?s′ ?p ?o* .
eq-rep-p	*?p* owl:sameAs *?p′* . *?s ?p ?o* .	*?s ?p′ ?o* .
eq-rep-o	*?o* owl:sameAs *?o′* . *?s ?p ?o* .	*?s ?p ?o′* .

Table 17. OWL 2 RL/RDF rules containing no T-atoms, but multiple A-atoms in the body—all relate to supporting the positive semantics of `owl:sameAs`, and all give quadratic materialisation

$\mathsf{TBody}(R) \neq \emptyset$ and $\lvert\mathsf{ABody}(R)\rvert > 1$			
ID	**Body**		**Head**
	terminological	*assertional*	
linear materialisation w.r.t. assertional data			
cls-int1	*?c* owl:intersectionOf (***?c***$_1$ ... ***?c***$_n$) .	*?y* a *?c*$_1$, ... , *?c*$_n$.	*?y* a *?c* .
cls-svf1	*?x* owl:someValuesFrom ***?y*** ; owl:onProperty ***?p*** .	*?u ?p ?v* . *?v* a *?y* .	*?u* a *?x* .
cls-avf	***?x*** owl:allValuesFrom *?y* ; owl:onProperty ***?p*** .	*?u ?p ?v* ; a *?x* .	*?v* a *?y* .
quadratic materialisation w.r.t. assertional data			
prp-fp	***?p*** a owl:FunctionalProperty .	*?x ?p ?y*$_1$, *?y*$_2$.	*?y*$_1$ owl:sameAs *?y*$_2$.
prp-ifp	***?p*** a owl:InverseFunctionalProperty .	*?x*$_1$ *?p ?y* . *?x*$_2$ *?p ?y* .	*?x*$_1$ owl:sameAs *?x*$_2$.
prp-key	***?c*** owl:hasKey (***?p***$_1$ ... ***?p***$_n$)	*?x ?p*$_1$ *?z*$_1$; ... ; *?p*$_n$ *?z*$_n$, a *?c* . *?y ?p*$_1$ *?z*$_1$; ... ; *?p*$_n$ *?z*$_n$, a *?c* .	*?x* owl:sameAs *?y* .
cls-maxc2	***?x*** owl:maxCardinality 1 ; owl:onProperty ***?p*** .	*?u* a *?x* ; *?p ?y*$_1$, *?y*$_2$.	*?y*$_1$ owl:sameAs *?y*$_2$.
cls-maxqc3	***?x*** owl:maxQualifiedCardinality 1 . ***?x*** owl:onProperty ***?p*** ; owl:onClass ***?c*** .	*?u* a *?x* ; *?p ?y*$_1$, *?y*$_2$. *?y*$_1$ a *?c* . *?y*$_2$ a *?c* .	*?y*$_1$ owl:sameAs *?y*$_2$
cls-maxqc4	***?x*** owl:maxQualifiedCardinality 1 . owl:onProperty ***?p*** ; owl:onClass owl:Thing .	*?u* a *?x* ; *?p ?y*$_1$, *?y*$_2$.	*?y*$_1$ owl:sameAs *?y*$_2$
prp-trp	***?p*** a owl:TransitiveProperty .	*?x ?p ?y* . *?y ?p ?z* .	*?x ?p ?z*
prp-spo2	*?p* owl:propertyChainAxiom (***?p***$_1$ ... ***?p***$_n$) .	*?u*$_1$ *?p*$_1$ *?u*$_2$. *?u*$_2$ *?p*$_2$ *?u*$_3$ . ... *?u*$_n$ *?p*$_n$ *?u*$_{n+1}$.	*?u*$_1$ *?p ?u*$_{n+1}$.

Table 18. OWL 2 RL/RDF rules containing some T-atoms and multiple A-atoms in the body

Head(R) = $\bot$		
ID	**Body**	
	terminological	*assertional*
eq-diff1	-	*?x* owl:sameAs *?y* . *?x* owl:differentFrom *?y* .
eq-diff2	-	*?x* a owl:AllDifferent ; owl:members (*?z*$_1$...*?z*$_n$) . *?z*$_i$ owl:sameAs *?z*$_j$. (*i*≠*j*)
eq-diff3	-	*?x* a owl:AllDifferent ; owl:distinctMembers (*?z*$_1$...*?z*$_n$) . *?z*$_i$ owl:sameAs *?z*$_j$. (*i*≠*j*)
prp-irp	***?p*** a owl:IrreflexiveProperty .	*?x ?p ?x* .
prp-asyp	***?p*** a owl:AsymmetricProperty	*?x ?p ?y* . *?y ?p ?x* .
prp-pdw	***?p***$_1$ owl:propertyDisjointWith ***?p***$_2$.	*?x ?p*$_1$ *?y* ; *?p*$_2$ *?y* .
prp-adp	*?x* a owl:AllDisjointProperties ; owl:members (***?p***$_1$...***?p***$_n$) .	*?u ?p*$_i$ *?y* ; *?p*$_j$ *?y* . (*i*≠*j*)
prp-npa1	-	*?x* owl:sourceIndividual *?i*$_1$. *?x* owl:assertionProperty *?p* . *?x* owl:targetIndividual *?i*$_2$. *?i*$_1$ *?p ?i*$_2$.
prp-npa2	-	*?x* owl:sourceIndividual *?i* . *?x* owl:assertionProperty *?p* . *?x* owl:targetValue *?lt* . *?i ?p ?lt* .
cls-nothing2	-	*?x* a owl:Nothing .
cls-com	***?c***$_1$ owl:complementOf ***?c***$_2$.	*?x* a *?c*$_1$, *?c*$_2$.
cls-maxc1	***?x*** owl:maxCardinality 0 ; owl:onProperty ***?p*** .	*?u* a *?x* ; *?p ?y* .
cls-maxqc1	***?x*** owl:maxQualifiedCardinality 0 ; owl:onProperty ***?p*** ; owl:onClass ***?c*** .	*?u* a *?x* ; *?p ?y* . *?y* a *?c* .
cls-maxqc2	***?x*** owl:maxQualifiedCardinality 0 ; owl:onProperty ***?p*** ; owl:onClass owl:Thing .	*?u* a *?x* ; *?p ?y* .
cax-dw	***?c***$_1$ owl:disjointWith ***?c***$_2$.	*?x* a *?c*$_1$, *?c*$_2$.
cax-adc	*?x* a owl:AllDisjointClasses ; owl:members (***?c***$_1$...***?c***$_n$) .	*?z* a *?c*$_i$, *?c*$_j$. (*i*≠*j*)
dt-not-type	-	*?lt* a *?dt* . (s.t. *?lt* is an ill-typed literal)

Table 19. OWL 2 RL/RDF "constraint" rules

ID	partially covered by recursive rule(s)
scm-cls	*incomplete* for `owl:Thing` membership inferences[a]
scm-sco	cax-sco
scm-eqc1	cax-eqc1, cax-eqc2
scm-eqc2	cax-sco
scm-op	*no unique assertional inferences*
scm-dp	*no unique assertional inferences*
scm-spo	prp-spo1
scm-eqp1	prp-eqp1, prp-eqp2
scm-eqp2	prp-spo1
scm-dom1	prp-dom, cax-sco
scm-dom2	prp-dom, prp-spo1
scm-rng1	prp-rng, cax-sco
scm-rng2	prp-rng, prp-spo1
scm-hv	prp-rng, prp-spo1
scm-svf1	*incomplete:* cls-svf1, cax-sco
scm-svf2	*incomplete:* cls-svf1, prp-spo1
scm-avf1	*incomplete:* cls-avf, cax-sco
scm-avf2	*incomplete:* cls-avf, prp-spo1
scm-int	cls-int2
scm-uni	cls-uni

Table 20. Enumeration of the coverage of inferences in case of the omission of rules in Table 14 wrt. inferencing over assertional knowledge by recursive application of rules in Table 16: underlined rules are not supported, and thus we would encounter incompleteness wrt. assertional inference (would not affect a full OWL 2 RL/RDF reasoner which includes the underlined rules).

[a] In our scenario, are not concerned—we filter out such statements and rules such as cls-svf2 and cls-maxqc2 encode direct support for `owl:Thing`.

B CURIE Prefixes Used

Herein, we enumerate the CURIE prefixes [**?**] used throughout this tutorial to abbreviate URIs.

Prefix	**URI**
aifb:	http://www.aifb.kit.edu/id/
b2r2008:	http://bio2rdf.org/bio2rdf-2008.owl#
contact:	http://www.w3.org/2000/10/swap/pim/contact#
dc:	http://purl.org/dc/elements/1.1/
dct:	http://purl.org/dc/terms/
doap:	http://usefulinc.com/ns/doap#
ex*:	*arbitrary example namespace*
foaf:	http://xmlns.com/foaf/0.1/
frbr:	http://purl.org/vocab/frbr/core#
geonames:	http://www.geonames.org/ontology#
mo:	http://purl.org/ontology/mo/
opiumfield:	http://rdf.opiumfield.com/lastfm/spec#
owl:	http://www.w3.org/2002/07/owl#
po:	http://purl.org/ontology/po/
plink:	http://buzzword.org.uk/rdf/personal-link-types#
pres:	http://www.w3.org/2004/08/Presentations.owl#
rail:	http://ontologi.es/rail/vocab#
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
skos:	http://www.w3.org/2004/02/skos/core#
wgs84:	http://www.w3.org/2003/01/geo/wgs84_pos#
wn:	http://xmlns.com/wordnet/1.6/

Table 21. Used prefixes