NUI Galway
OÉ Gaillimh

Exploiting RDFS and OWL for Integrating Heterogeneous, Large-Scale, Linked Data Corpora

Aidan Hogan

Supervisor: **Dr. Axel Polleres**
Internal Examiner: **Prof. Stefan Decker**
External Examiner: **Prof. James A. Hendler**

Dissertation submitted in pursuance of the degree of Doctor of Philosophy

Digital Enterprise Research Institute, Galway
National University of Ireland, Galway / Ollscoil na hÉireann, Gaillimh

April 11, 2011

The research presented herein was supported by an IRCSET Postgraduate Scholarship and by Science Foundation Ireland under Grant No. SFI/02/CE1/I131 (Lion) and Grant No. SFI/08/CE/I1380 (Lion-2).

"If you have an apple and I have an apple and we exchange these apples then you and I will still each have one apple. But if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas."

—George Bernard Shaw

Acknowledgements

First, thanks to the taxpayers for the pizza and (much needed) cigarettes;

...thanks to friends and family;

...thanks to the various students and staff of DERI;

...thanks to the URQ folk;

...thanks to people with whom I have worked closely, including Alex, Antoine, Jeff, Luigi and Piero;

...thanks to people with whom I have worked *very* closely, particularly Andreas and Jürgen;

...thanks to John and Stefan for the guidance;

...thanks to Jim for the patience and valuable time;

...and finally, a big thanks to Axel for everything.

Abstract

The Web contains a vast amount of information on an abundance of topics, much of which is encoded as structured data indexed by local databases. However, these databases are rarely interconnected and information reuse across sites is limited. Semantic Web standards offer a possible solution in the form of an agreed-upon data model and set of syntaxes, as well as metalanguages for publishing schema-level information, offering a highly-interoperable means of publishing and interlinking structured data on the Web. Thanks to the Linked Data community, an unprecedented lode of such data has now been published on the Web—by individuals, academia, communities, corporations and governmental organisations alike—on a medley of often overlapping topics.

This new publishing paradigm has opened up a range of new and interesting research topics with respect to how this emergent "Web of Data" can be harnessed and exploited by consumers. Indeed, although Semantic Web standards theoretically enable a high level of interoperability, *heterogeneity* still poses a significant obstacle when consuming this information: in particular, publishers may describe analogous information using different terminology, or may assign different identifiers to the same referents. Consumers must also overcome the classical challenges of processing Web data sourced from multitudinous and unvetted providers: primarily, *scalability* and *noise*.

In this thesis, we look at tackling the problem of heterogeneity with respect to consuming large-scale corpora of Linked Data aggregated from millions of sources on the Web. As such, we design bespoke algorithms—in particular, based on the Semantic Web standards and traditional Information Retrieval techniques—which leverage the declarative schemata (a.k.a. terminology) and various statistical measures to help smooth out the heterogeneity of such Linked Data corpora in a *scalable* and *robust* manner. All of our methods are distributed over a cluster of commodity hardware, which typically allows for enhancing performance and/or scale by adding more machines.

We first present a distributed crawler for collecting a generic Linked Data corpus from millions of sources; we perform an open crawl to acquire an evaluation corpus for our thesis, consisting of 1.118 billion facts of information collected from 3.985 million individual documents hosted by 783 different domains. Thereafter, we present our distributed algorithm for performing a links-based analysis of the data-sources (documents) comprising the corpus, where the resultant ranks are used in subsequent chapters as an indication of the importance and trustworthiness of the information they contain. Next, we look at custom techniques for performing rule-based materialisation, leveraging RDFS and OWL semantics to infer new information, often using mappings—provided by the publishers themselves—to translate between different terminologies. Thereafter, we present a formal framework for incorporating metainformation—relating to trust, provenance and data-quality—into this inferencing procedure; in particular, we derive and track ranking values for facts based on the sources they originate from, later using them to repair identified noise (logical inconsistencies) in the data. Finally, we look at two methods for consolidating coreferent identifiers in the corpus, and we present an approach for discovering and repairing incorrect coreference through analysis of inconsistencies. Throughout the thesis, we empirically demonstrate our methods against our real-world Linked Data corpus, and on a cluster of nine machines.

Declaration

I declare that this thesis is composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Aidan Hogan
April 11, 2011

Contents

Chapter 1

Introduction

"Getting information off the Internet is like taking a drink from a fire hydrant."

—Mitchell Kapor

Telecommunication has inarguably come a long way since the days of smoke signals and carrier pigeons. Today, one quarter of the world's population is now connected by the *Internet*: a shared global networking infrastructure enabling near-seamless communication across (most) geopolitical divides.[1] Such unprecedented fluency in inter-communication and information dissemination—between businesses, governments, academia, organisations and individuals—has had profound socio-economic impact in an exceptionally short time-span. In particular, the advent of the *World Wide Web* [Berners-Lee and Fischetti, 1999] has enabled public publishing and consumption of information on a unique scale: publishing on the Web is low-cost and largely accessible, with the size of the potential audience limited only by demand for the content.

As a result, the Web consists of at least 41 billion unique documents [Lee et al., 2008], forming a vast mosaic of information on a veritable plethora of topics. However, what's a person to do with 41 billion documents? Humans not only need machines to store, transmit and display information, but also to characterise, categorise, prioritise and generally organise information for their convenient consumption; however, most of the 41 billion documents encode their information in human readable prose, the meaning of which is largely inscrutable to machines. Despite this, search engines—such as the now ubiquitous Google engine [Brin and Page, 1998]—service tens of billions of keyword queries per month, leveraging the content and limited structure of the indexed documents (for example: title, hyperlink, etc.) and statistics derivable from the corpus as a whole to identify and return a prioritised list of query-relevant documents.[2] Such engines act like a panoptic (but largely illiterate) librarian, pointing users to relevant reading material—possibly several documents—from which the required information is most likely to be gleaned: the gleaning, thereafter, is up to the user.

However, the library analogy is only representative of an antediluvian Web centred around static documents: recent advancements in related technologies have changed how we view and interact with the Web, culminating in a flood of user-generated data. Dubbed *Web 2.0* [O'Reilly, 2006], more flexible client/server communication has lead to more interactive sites, eventually blurring the classical roles of publisher and consumer: the casual Web user no longer just passively consumes information, but instead participates in a more social Web [Breslin et al., 2009, § 3]—for example, generating rich meta-content (ratings, comments,

[1] See `http://www.internetworldstats.com/stats.htm`; retr. 2010/10/02.

[2] As Halevy et al. [2009] put forward, simple statistical models (such as n-gram analysis) applied over vast unstructured or semi-structured corpora (as readily derivable from the Web) are often sufficient for machines to "emulate" an understanding of the meaning of data and solve complex tasks without requiring formalised knowledge.

shares, tags, bookmarks, links, etc.) while browsing. This has had two major side-effects.

Firstly, more and more structured content is appearing on the Web, with pages being dynamically generated depending on a specific user request or browsing action: information is no longer stored away in static documents, but instead stored in structured indices—typically relational databases—used to power individual sites [Ghanem and Aref, 2004]. Thus the primary function of documents is no longer as containers of information, but as interfaces for conveying or requesting information; as such, documents constitute the façade of the current Web, with more and more rich content (instead) being stored in structured databases.

Secondly, user-generated content abides by a form of preferential attachment [Barabási and Albert, 1999]: the more user-generated content a given site attracts, the more users and user loyalty that site will attract. Thus, content on the Web coalesces into archipelagos of information, where large sites house massive corpora of user-generated content, but where remote sites can only intercommunicate or share their data based on ad-hoc mappings or alignments. Indeed, the underlying data of many sites constitute their competitive advantage: although free, low-volume access is inherently made available through the defined public services, very few sites offer flexible or high-volume access to their underlying corpus. Thus, for example, users find that their data are "locked in" by different sites, can only be interacted with in accordance with that site's functionality, and typically cannot be repurposed (e.g., imported into another site with different functionality).

For many commercial sites, providing open access to their underlying corpus would encourage competitors and redirect traffic to other interfaces over their data; such sites would clearly be reluctant to do so without some alternative economic impetus. However, non-commercial sites—such as governmental, academic or community-run sites—typically host data under liberal licenses, and commercial sites may make some subset of their data available in order to attract users or traffic (e.g., by encouraging third-party applications over the data which ultimately direct traffic back to the originating site).

Looking towards the future, a major question is thus: how can structured data be published on the Web such that they can be accessed, parsed and interpreted in a standardised way (i.e., assuming a site had the motivation to do so, how can they publish their underlying corpus in a manner which best facilitates public reuse), and how can the structured data of different independent sites be interlinked in order to enable post-hoc discovery of relevant information across the Web (e.g., how can two sites with overlapping subject matter interlink their corpora in a meaningful way)?

The *Semantic Web* community [Berners-Lee, 1998b] is (sometimes indirectly) taking *tentative* steps towards answering these questions; centring around the Resource Description Framework (RDF) [Manola et al., 2004] for publishing data in a machine-readable format, the Semantic Web offers a standardised means of representing information on the Web such that:

1. resources—anything with identity—can be named using Unique Resource Identifiers (URIs);

2. data about resources are encoded within a standard, flexible data-model using standard syntaxes and thus are structurally uniform and can be parsed and handled through standard APIs and applications;

3. the meaning of data can be encoded using declarative schemata (or ontologies) represented in RDF, whose semantics can be defined within the RDF Schema (RDFS) [Brickley and Guha, 2004] and Web Ontology Language (OWL) [Hitzler et al., 2009] standards.

Semantic Web standards focus on interoperability across data sources, on both a syntax, data (assertional) and schemata (terminological) level, where Uniform Resource Identifiers (URIs) [Berners-Lee et al., 2005] offer a global naming scheme compatible with current Web technologies, RDF provides a standard structured data model for describing those named resources, and RDFS and OWL provide a standard means of declaratively publishing schema-level information used to describe those resources.

As such, the Semantic Web community has addressed many technical and research challenges (sometimes) regarding the dissemination of open, structured data on the Web. As a means of promoting grass-roots adoption of Semantic Web standards, the *Linked Data* community [Bizer et al., 2009a] have advocated a set of best principles for collaboratively publishing and interlinking structured data over the Web, as follows:

1. *use URIs* as names for things;
2. *use HTTP URIs* so those names can be looked up (aka. *dereferencing*);
3. *return useful information* upon lookup of those URIs (esp. RDF);
4. *include links* by using URIs which dereference to remote documents.

—paraphrasing Berners-Lee [2006]

The result can be conceptualised as a *Web of Data*, where URIs identify *things*, dereferencing URIs (by means of the Hypertext Transfer Protocol [Fielding et al., 1999]) returns structured (RDF) information about those things, and that structured information is inherently composed of related URIs which constitute links to other sources enabling further discovery.

This vision is currently being realised: 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 [Kobilarov et al., 2009], BestBuy [Hepp, 2009], New York Times[3], Freebase[4]), community-driven efforts (e.g., WIKIPEDIA/DBpedia[5] [Bizer et al., 2009b]), 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 Numbers[13] [Vrandečíc et al., 2010]). 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.

As such, there now exists a rich vein of heterogeneous, structured and interlinked data on the Web. Given the success of current warehousing approaches for largely unstructured Web data—such as the aforementioned Google search engine—one might question what is further possible for structured data. Taking this cue, herein we discuss algorithms and analyses which exploit the inherent semantics and statistics of RDF data to perform fully-automatic, scalable and distributed alignment of Linked Data exports provided by multitudinous, independent publishers, creating a more coherent corpus such that can be exploited by—in particular—emerging Linked Data warehousing systems.

1.1 Problem Statement

As we have discussed, Semantic Web standards aim at providing the common substrate needed for publishing highly-interoperable data on the Web, founded upon the RDF data-model. Subsequently, Linked Data principles provide an agreed-upon method for publishing RDF on the Web where dereferencable URIs offer a

[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

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

simple means of locating structured data about a given resource, which contain links for discovery of further data.

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 naming ensures that all available data about the same resource can be aligned across all sources, and 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 [Barabási and Albert, 1999]—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 identifiers and terminology needed to fully realise the Web of Data is probably infeasible.

1.1.1 Incomplete Agreement on Assertional Identifiers

Complete agreement upon a single URI for each possible resource of interest is unrealistic, and would require either a centralised naming registry to corroborate name proposals, or agreement upon some universal bijective naming scheme compatible with any arbitrary resource. Although partial (and significant) agreement on ad-hoc URIs is more feasible, there is also an inherent conflict between encouraging reuse of identifiers and making those identifiers dereferenceable: a publisher reusing an external URI to identify some *thing* waives the possibility of that URI dereferencing to her local contribution.

Thus, although in theory Linked Data can be arbitrarily merged and heterogeneous data about a common resource will coalesce around a common identifier, in practice, common identifiers are not always feasible, or possibly even desirable. Consequently, we propose that Linked Data needs some means of (i) *resolving coreferent identifiers* which signify the same thing; (ii) *canonicalising coreferent identifiers* such that consumers can access and process a heterogeneous corpus as if (more) complete agreement on identifiers was present. Without this, the information about all resources in the Linked Data corpus will be fractured across naming schemes, and a fundamental goal of the Web of Data—to attenuate the traditional barriers between data publishers—will be compromised.

1.1.2 Use of Analogous Terminologies

Similarly, 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 [Bizer et al., 2008].

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—e.g., in a simple case, formulate a disjunctive (sub-)query for triples using either of the `foaf:made`/`foaf:maker` properties. Consequently, we propose that Linked Data needs some means of *translating between terminologies* to enable more complete query-answering (in the general case).[14]

1.2 Hypothesis

There has, of course, been recognition of the above stated problems within the Linked Data community; publisher-side "solutions" involving RDFS and OWL semantics have been proposed. Firstly, in [Bizer et al., 2008, § 6]—a tutorial positioned as the "definitive introductory resource" to Linked Data on the prominent `linkeddata.org` site—Bizer et al. state that `owl:sameAs` should be used to interlink coreferent resources in remote datasets:

> *"It is common practice to use the owl:sameAs property for stating that another data source also provides information about a specific non-information resource."*
>
> **—Bizer et al. [2008, § 6]**

Thus, the `owl:sameAs` property can be used to relate locally defined (and ideally dereferenceable) identifiers to external legacy identifiers which signify the same thing. This approach offers two particular advantages: (i) publishers can define an ad-hoc local naming scheme for their resources—thus reducing the initial inertia for Linked Data publishing—and thereafter, incrementally provide mappings to external coreferent identifiers as desirable; (ii) multiple dereferenceable identifiers can implicitly provide alternative sources of information for a given resource, useful for discovery.

Furthermore, OWL provides the class `owl:InverseFunctionalProperty`: properties contained within this class have values unique to a given resource—loosely, these can be thought of as *key* values where two resources sharing identical values for some such property are, by OWL semantics, coreferent. Along these lines, inverse-functional properties can be used in conjunction with existing identification schemes—such as ISBNs for books, EAN·UCC-13 or MPN for products, MAC addresses for network-enabled devices, etc.—to bootstrap identity on the Web of Data within certain domains; such identification values can be encoded as simple datatype strings, thus bypassing the requirement for bespoke agreement or mappings between URIs. Also, "information resources" with indigenous URIs can be used for (indirectly) identifying related resources, where examples include personal email-addresses, personal homepages, etc. Although Linked Data literature has not explicitly endorsed or encouraged such usage, prominent grass-roots efforts publishing RDF on the Web rely (or have relied) on inverse-functional properties for maintaining consistent identity.[15] Similar other constructs are available in OWL for resolving coreference, such as `owl:FunctionalProperty`, `owl:cardinality`, and `owl:hasKey` (the latter was introduced in the updated OWL 2 standard). Note that these OWL constructs require agreement on terminology—for example, agreement on a given property term to denote the ISBN attribute—without which, coreference cannot be established.

As motivated before, we need some means of aligning the terminologies of different datasets. Along these lines, the Linked Data literature offers some guidelines on best practices with respect to vocabularies, as follows:

1. ***Do not define new vocabularies from scratch*** [...] *complement existing vocabularies with addi-*

[14] In particular, we wish to leverage existing *mappings* between internal and external terms, often included in the published vocabularies themselves; we do not address the means of generating mappings, but rather the means of *using* them. In our opinion, techniques from fields such as ontology matching [Jérôme Euzenat, 2007] have yet to prove themselves applicable for *consuming* Linked Data—particularly large, heterogeneous corpora—and we feel that such techniques have greater potential as a *publisher-side* technology for supervised discovery and maintenance of vocabulary mappings.

[15] For example, in the Friend Of A Friend (FOAF) community—a vocabulary and associated project dedicated to disseminating personal profiles in RDF—a technique called smushing was proposed to leverage such properties for identity, serving as an early precursor to methods described herein (see `http://wiki.foaf-project.org/w/Smushing`; retr. 2011/01/22).

tional terms [...].

2. ***Provide for both humans and machines.*** [...] *Don't forget to add prose, e.g. rdfs:comments for each term invented. Always provide a label for each term using the rdfs:label property.*

3. ***Make term URIs dereferenceable.*** [...]

4. ***Make use of other people's terms.*** [...] *Common properties for providing such mappings are rdfs:subClassOf or rdfs:subPropertyOf.*

5. ***State all important information explicitly.*** *For example, state all ranges and domains explicitly.* [...] *Don't leave important information out!*

6. ***Do not create over-constrained, brittle models; leave some flexibility for growth.*** *For instance, if you use full-featured OWL to define your vocabulary, you might state things that lead to unintended consequences and inconsistencies when somebody else references your term in a different vocabulary definition. Therefore, unless you know exactly what you are doing, use RDF-Schema to define vocabularies.*

—excerpts verbatim from Bizer et al. [2008, § 6]

Thus, by item (1) and (4), we would expect terminological vocabularies to reuse and provide mappings to external vocabularies. We would also expect some lightweight semantics to be defined by point (5), although we note that "full-fledged" OWL—and in fact OWL itself—is discouraged for "inexperienced publishers" (our words) in point (6).

Thus, our hypothesis can be summarised as follows:

> **Given a heterogeneous Linked Data corpus, the RDFS and OWL semantics of the vocabularies it contains can be (partially) leveraged in a domain-agnostic, scalable, Web-tolerant manner for the purposes of (i) automatically translating between (possibly remote) terminologies; and (ii) automatically resolving (possibly remote) coreferent assertional identifiers.**

There are three requirements implicit in the above hypothesis, the combination of which formulate the novelty of this thesis:[16]

- **domain-agnosticism**: the methods presented herein rely on a-priori knowledge derived from the Semantic Web standards themselves (RDF(S)/OWL (2)) and the core Linked Data principles—prior domain- or vocabulary-specific knowledge could be considered biased, inflexible, infeasible to create/-maintain, etc., and thus is not required by our methods;

- **scalability**: the methods presented herein aim at near-linear scale (more accurately, $O(n \log n)$ linearithmic scale), where we wish to apply our methods over Linked Data corpora collected from millions of sources;

- **Web-tolerance**: the methods presented herein are designed to operate over arbitrarily sourced Linked Data, which requires tolerance to noisy, conflicting, inconsistent, and generally impudent contributions from unknown publishers.

[16] More explicitly, our goal herein is to investigate the degree to which—and the respective means by which—the two aims of the hypothesis can be achieved within the bounds of the three requirements in the hypothesis.

Although the above requirements have been analysed in isolation by related works in the literature—which will be discussed as pertinent—the combination of all three have received little attention, and we propose that the above requirements capture the realistic scenario faced by consumers of large-scale arbitrary Linked Data corpora (e.g., Linked Data warehouses).

1.3 Contribution and Thesis Structure

This thesis aims to demonstrate potential use-cases for RDFS/OWL semantics found within Linked Data, with focus on scale and Web-tolerance.

With respect to domain-agnosticism, our methods do not require any input other than some Linked Data corpus and knowledge about HTTP redirects encountered during acquisition (used for tracking dereferenceability).

With respect to scale, we design, implement and evaluate distributed algorithms which can be applied over a cluster of commodity hardware (as commonly deployed by current warehousing systems) and which aim to provide *horizontal scaling*: adding more machines increases the total amount of data that can be processed (increases scale), and/or decreases the total execution time for similar scale (increases efficiency). We also focus on lightweight algorithms implementable by means of simple sorts and file-scans. Thus, for reasons of scalability/computational feasibility, some of our methods are deliberately incomplete, and do not exploit all of the possible RDFS/OWL semantics.

With respect to Web-tolerance, we propose some complementary techniques which incorporate a notion of provenance—the source of data—to curtail the effects of unwanted third-party contributions. Given the possibility of noisy or incorrect data, we also incorporate algorithms which cross-check the results of our methods, and attempt to identify, diagnose, and repair problems arising from application thereof.

We thus begin by describing a distributed method for *crawling* Linked Data, which we use for deriving a test-bed corpus—this corpus contains 1.118 billion Linked Data statements crawled from 3.985 million RDF/XML sources and is used throughout the rest of the thesis for evaluation. We also describe a distributed method for *ranking* Linked Data documents, which uses a PageRank [Page et al., 1998] inspired links-based analysis to assign the sources comprising the corpus a score reflecting their (Eigenvector) centrality—these ranking score are used in later chapters for analysing the importance of pertinent RDFS and OWL primitives on the Web of Data, as well as being used to assign ranks to individual triples.

Thereafter, we propose our method for performing *reasoning*. We first propose a scalable, distributed implementation which performs incomplete materialisation—asserting translations of the data into different combinations of terminology—with respect to a subset of OWL 2 RL/RDF rules. We provide various formal results relating to completeness, and describe and evaluate a number of optimisations for improving efficiency. We then propose a method for performing *authoritative reasoning*, which tracks the provenance of terminological data during reasoning and ignores unverifiable RDFS/OWL axioms. Additionally, we present a survey of the use of different terminological axioms in our corpus, giving insights into the most commonly used RDFS and OWL primitives on the Web of Data.

We then extend upon the given reasoning approach using an annotation framework, whereby triples can be *annotated* with additional meta-information which can be tracked and transformed during the reasoning process. We use this annotation framework to track the above notions of authority during reasoning. Taking the ranks of documents as input, we propagate ranking scores to individual triples, indicating some level of (loosely) "trust" assigned to a given triple. Based on the ranks of the input triples, the annotated reasoning framework then produces a rank value for each triple inferred. We then apply inconsistency-detection over the merge of the input and materialised corpora to diagnose suspected publishing errors and unintended reasoning consequences; we repair the diagnosed errors using the rank measures and derive a parsimonious repair which defeats the “marginal-view”.

We then proceed by tackling the resolution of coreferent assertional identifiers in the corpus.[17] We begin with a baseline approach which operates over explicit `owl:sameAs` relations, and then subsequently extend the approach to consider a more complete RDFS/OWL semantics—for the latter, we reuse the above reasoning results to pre-align terminologies and thus derive more complete coreference information. We use this coreference information to *consolidate* the corpus: from each set of coreferent identifiers, we select a “canonical identifier” to identify the consolidated resource, and rewrite the data accordingly. We also describe a statistical method for deriving concurrence measures between resources we deem to be “similar” based on the number and nature of inlinks and outlinks they share. Finally, we check the consolidated data for inconsistencies—aiming to diagnose incorrect coreferences/consolidation—and sketch a process for repair based on (i) how the coreferences are derived, and (ii) the concurrence measure observed between the original resources involved.

Throughout, we provide detailed performance evaluation for application of our methods over a cluster of commodity hardware, and discuss the fecundity of our algorithms with respect to our Linked Data corpus.

The remainder of the thesis is structured as follows:

- Chapter 2 provides background on the World Wide Web, Semantic Web standards, and Linked Data publishing;
- Chapter 3 introduces some core concepts and notation used throughout the rest of the thesis;
- Chapter 4 describes our distributed Linked Data crawler, which we use to derive a test-bed corpus for evaluation of our work; we also describe our distributed ranking implementation for applying links-based analysis over the documents comprising the corpus;
- Chapter 5 details our approach to reasoning over Linked Data;
- Chapter 6 continues by introducing our annotation framework, including a lightweight method for repairing inconsistencies;
- Chapter 7 describes our method for resolving coreferences and thereafter consolidating Linked Data, as well as our method for detecting and revising any coreferences which instigate inconsistencies;
- Chapter 8 provides critical discussion of our work, suggestions for future directions, and concludes.

1.4 Impact

Parts of the work presented herein have been published in various international workshops, conferences, as well as a journal article, which we now briefly introduce in chronological order.

- we presented preliminary results for consolidating RDF Web data at the I3 Workshop [Hogan et al., 2007a], which serves as a precursor to work presented in Chapter 7;
- we presented preliminary reasoning results at the Asian Semantic Web Conference [Hogan et al., 2008], which serves as a precursor to the approach presented in Chapter 5;
- we published an extension of the above paper (with new algorithms, formalisms and extended experi-

[17] Note that herein, we deliberately decouple reasoning with respect to terminology, and coreference analysis of assertional identifiers; although OWL reasoning typically deals with both, we view them as intuitively distinct challenges which require different approaches, particularly for the Linked Data use-case.

mental results) in the International Journal of Semantic Web and Information Systems [Hogan et al., 2009a];

- we presented an extension of our reasoning approach to incorporate OWL 2 RL/RDF rules at the Web Rules and Reasoning conference [Hogan and Decker, 2009];
- we presented a proposal for a statistical method of deriving coreference at the New Forms of Reasoning for the Semantic Web workshop, co-located with ESWC [Hogan et al., 2010d]—this work forms the basis for the concurrence analysis presented in Chapter 7 (and Appendix D);
- more recently, regarding our reasoning algorithms, we published general completeness results and optimisations, distribution strategies, and new experimental performance analysis at the International Semantic Web Conference [Hogan et al., 2010c]—these results are presented in Chapter 5.

Much of the work presented herein has recently been submitted for review; in particular, we have submitted three papers for review to the Journal of Web Semantics. The first describes our work to-date on the Semantic Web Search Engine (SWSE)—a Linked Data warehouse which provides much of the inspiration for this thesis, and in which we detail our distribution architecture and crawling framework; we also discuss consolidation of Linked Data using explicit `owl:sameAs` and integration of the reasoning approach presented herein into the SWSE architecture [Hogan et al., 2010b]. The second submission relates to the annotation framework as presented in Chapter 6 [Bonatti et al., 2011]. The third submission relates to the results of our work on coreference, consolidation, etc., as presented in Chapter 7 [Hogan et al., 2010e].

Besides the above papers, we have also been involved in numerous other published works which have provided much inspiration for this thesis. Highlights include a paper presented at the Scalable Semantic Web Knowledge Base Systems workshop on links-based ranking of RDF Web data which integrates links between the source- and data-level graphs [Hogan et al., 2006]; an early overview of the SWSE system published in the demo proceedings of the World Wide Web conference [Hogan et al., 2007b]; our paper describing YARS2—a highly scalable, distributed quad-store and SPARQL engine—which we presented at the International Semantic Web Conference in 2007 [Harth et al., 2007]; a paper addressing the quality of RDF Web data which we presented at the Linked Data on the Web workshop, co-located with WWW [Hogan et al., 2010a]; and a position paper in the inaugural issue of the new Semantic Web Journal discussing future directions we feel to be important for Semantic Web research [Polleres et al., 2010].

Many of the above papers have been cited numerous times within the research community; e.g., see [Oren et al., 2009a; Bizer et al., 2009a; Cheng and Qu, 2009; Urbani et al., 2009; Weaver and Hendler, 2009; Franz et al., 2009; Neumann and Weikum, 2010; Urbani et al., 2010; Hitzler and van Harmelen, 2010] for some recent citations.

As previously mentioned, the baseline reasoning and consolidation approaches have been integrated into the SWSE engine, with a public prototype available at `http://swse.deri.org/`.

Chapter 2

Background

"Knowledge of what is does not open the door directly to what should be."

—Albert Einstein

2.1 The World Wide Web

The World Wide Web (or simply the Web) is a global system of interlinked documents accessible via the Internet. Remotely stored documents are furnished with a globally unique address—a Uniform Resource Locator (URL)—that encodes the location from which (and, to a certain extent, the means by which) that document can be retrieved across the Internet. Traditionally, these documents consist of *hypertext* [Nelson, 1965]—specified by means of the HyperText Markup Langauge (HTML)—which primarily contains formatted natural language, digital images, and other rendering instructions for the client's *browser application*: a tool for retrieving and displaying remote Web documents. Importantly, documents can *hyperlink* to other related documents, embedding the URLs of target documents into the body of text, allowing users to browse between related documents.

The World Wide Web arose from seminal work by Tim Berners-Lee while an employee with CERN in Geneva, Switzerland. In the early 80s, Berners-Lee began work on a hypertext documentation engine called ENQUIRE [Berners-Lee, 1980], which was a functional predecessor to the World Wide Web;[1] Berners-Lee was responding to the complex information and communication needs presented by the technical, collaborative enviroment of CERN [Berners-Lee, 1993]. The ENQUIRE system centred around "cards" as information resources about "nodes", which could refer to a person, a software module, etc., and which could be interlinked using a selection of relations, such as `made`, `includes`, `uses`, `describes`.

However—and although Berners-Lee recognised the potential of such a tool for collaborative information tasks [Berners-Lee, 1993]—the ENQUIRE system was limited to a local file-system, and was not naturally suited to such a task: besides the lack of a physical communication layer, the ENQUIRE system allowed for open editing and required heavy co-ordination to keep information up-to-date and cross-linkage consistent. In order to provide a more open, collaborative tool, Berners-Lee started a new project called the World Wide Web:

"I wanted [ENQUIRE] *to scale so that if two people started to use it independently, and later*

[1] The ENQUIRE system itself has roots in other works such as Bush's hypothetical Memex [Bush, 1945], Engelbart's H-LAM/T system [Engelbart, 1962] and its successor NLS [Engelbart, 1968], and Nelson's work on hypertext and Xanadu [Nelson, 1965]; it's perhaps worth noting that Berners-Lee does not cite these works as direct influences on ENQUIRE or the Web [Berners-Lee, 1993].

> *started to work together, they could start linking together their information without making any other changes. This was the concept of the Web."*
>
> **—Berners-Lee [1993]**

By late 1990, Berners-Lee had developed initial versions of the technologies underpinning today's Web: the HyperText Markup Language (HTML) used for encoding document formatting and layout, the HyperText Trasfer Protocol (HTTP) for client/server communication and transmission of data—particularly HTML—over the Internet, the first Web client software (a "browser" called WorldWideWeb), and software to run the first Web server.

The first application of the Web was to make a browsable version of the CERN phone-book accessible to all employees' terminals; subsequently, the technology was adopted for organising the library mainframe of the Stanford Linear Accelerator Center, and thereafter, adoption of the Web grew amongst the High Energy Physics community [Berners-Lee, 1993]. Eventually, graphical Web browsers for various platforms were developed; a seminal browser released for the Windows Operating System, called Mosaic, quickly became popular due to its ease of installation and intuitive graphical interface. Fighting off some competition from the more established but more rigidly structured Gopher protocol [Anklesaria et al., 1993], the Web became more and more mainstream.

With the advent and rapid adoption of server-side scripting languages, new technologies for creating dynamic and interactive content quickly became widespread in the next years: the Common Gateway Interface standard, and later dedicated languages such as PHP Hypertext Preprocessor (PHP), Active Server Pages (ASP) and JavaServer Pages (JSP), enabled developers to provide more and more dynamic content through their server. Relational databases became increasingly popular to store structured information relating to the sites' content; a popular combination—dubbed "LAMP" and still in widespread use—combines the non-commercial Linux operating system, the Apache HTTP Server, the MySQL database management system, and one or more of the PHP/Perl/Python scripting languages.

Later advancements in client-side software saw website functionality begin to emulate that of desktop applications: key enabling technologies included client-side Javascript used to manipulate retrieved documents and information, Macromedia/Adobe Flash for incorporating multimedia (videos, animations, and other interactive artefacts) into webpages, Asynchronous JavaScript and XML (AJAX) enabling flexible asynchronous communication between client and server whilst the former interacts with a given page, and finally new Web browsers with built-in support of these burgeoning technologies.

As discussed at the outset, these technologies (amongst others) granted users much more participation in sites, posting comments, ratings or even primary content, leading to websites accumulating massive amounts of user-generated content. Various other publishing/communication paradigms also became common, including *blogging*: published entries listed in reverse-chronological order; *micro-blogging*: a simple means of broadcasting short messages to interested parties; and *wikis*: flexible, collaboratively editable pages. Other media further blurred the lines between publishing and communication; social networking sites, such as the prominent Facebook and MySpace services, allow friends to communicate and share content, but also to publish these contributions for a select audience to view.

Many sites—such as the aforementioned social networking sites, video hosting site YouTube, the Internet Movie Database (IMDb), the collaborative encyclopaedia WIKIPEDIA—have become hugely popular as a service to submit, exchange, curate, and interact with large corpora of user-generated content, typically in a highly collaborative and social environment.

This brings us to the Web we know today: a highly dynamic, highly flexible platform for hosting, publishing, adapting, submitting, interchanging, curating, editing and communicating various types of content, where many sites boast large corpora of rich user-generated data—typically stored in relational databases—

but where the content of different sites is primarily interconnected by generic hyperlinks.

2.2 The Semantic Web

The Web has inarguably been tremendously successful, and begs the question: *what's next*?

To begin to meaningfully answer this question, one has to look at the shortcomings of the current Web; along these lines, consider researching the question: *Which five universities have the highest number of living alumni who have been appointed or elected into the United States Congress (Senate or House of Representatives)?* One could hope that: (i) someone has previously performed this task and published their results, or (ii) a domain-specific site has the data and the functionality required to answer this query directly, or (iii) a domain-specific site has the data available for download in a structured format processable off-line; however, clearly these solutions do not extend to the general case.[2]

Assuming the above solutions don't apply—and from the user's experience, she knows that the data are undoubtedly on the Web—the task will likely require a large manual effort. Firstly, she may have to cross-reference published lists of university alumni, senators and representatives—likely from different sources—which should be easy enough to locate. She'll also have to find a reliable source of information about deaths, possibly from available news reports or online biographies. The resulting data may be unstructured or in heterogeneous formats: the user requires data in some consistent structured form as she wants to use some local software to perform the arduous cross-referencing necessary for the task, and so will have to apply some screen-scraping or other data extraction techniques to get what she needs. Once the data are in a computer-processable state, the user will quickly run into problems with peoples names: titles may not match across sources, abbreviations may be used, different people may share the same name, etc.

One can of course imagine variations on the above theme: original research which requires various levels of cross-referencing of various Web documents. Such tasks require: (i) structured data to be made available by the respective sources such that they can be subsequently processed (either client or server side) by machine; (ii) some means of resolving the identity of resources involved such that consistent cross-referencing can be performed.

Acknowledging such requirements, Berners-Lee [1998b] proposed the Semantic Web as a variation—or perhaps more realistically, an augmentation—of the current Web such that it is more amenable to machine-processing, and such that software agents can accomplish many of the tasks users must currently perform manually.

2.2.1 Resource Description Framework

The first major step towards realising this machine-processable Web came in early 1999 when the initial Resource Description Framework (RDF) became a W3C Recommendation [Lassila and Swick, 1999]. RDF provides a standardised means for expressing information such that it can be exchanged between RDF-aware agents without loss of meaning [Manola et al., 2004].

Notably, RDF is (implicitly) based on two major premises:

1. the *Open World Assumption* (OWA), which assumes that anything not known to be true is unknown, and not necessarily false as would be assumed in closed systems;
2. ***no*** *Unique Name Assumption* (UNA), which means that RDF does *not* assume that a name (in

[2] We note that Richard MacManus has defined a similar litmus test for the Semantic Web called "The Modigliani Test", whereby a lay user can get answers (in a structured format) to the question "tell me the locations of all the original paintings of Modigliani"—a moderately obscure artist from the early 20th century; see `http://www.readwriteweb.com/archives/the_modigliani_test_semantic_web_tipping_point.php` (retr. 2010/01/22).

particular, a URI) signifies something unique—more precisely, the mapping from names to things they identify is not assumed to be injective.

The standards built on top of RDF also (typically[3]) hold true to these premises. Given that RDF is intended for deployment on the Web, the OWA necessarily assumes that data are naturally incomplete, and the lack of UNA allows publishers to potentially identify the same thing using different identifiers, thus avoiding the need for a centralised naming service or some such.

Thereafter, RDF allows for describing *resources*—anything with discernible identity [Manola et al., 2004]—as follows:

1. resources are optionally defined to be members of *classes*, which are referenceable collections of resources—typically sharing some intuitive commonality—such that classes can themselves be described as resources;

2. resources are defined to have *values* for named *properties*; properties can themselves be described as resources, and values can be either:

    - a *literal value* representing some character-string, which can be optionally defined with either:
        - a *language tag*—for example, `en-IE`—denoting the language a prose-text value is written in; or
        - a *named datatype*—for example, a date-time datatype—which indicates a predefined primitive type with an associated syntax, means of parsing, and value interpretation;
    - a *resource*, indicating a directed, named relationship between the two resources.

RDF data of the above form can be specified by means of triples, which are tuples of the form:

$$(subject, predicate, object)$$

which, as aforementioned, can be used to designate classes to resources:

$$(\texttt{Fred}, \texttt{type}, \texttt{Employee})$$

to define literal-valued attributes of resources:

$$(\texttt{Fred}, \texttt{age}, \texttt{"56"\^{}\^{}xsd:int})$$

and/or to define directed, named relationships between resources:

$$(\texttt{Fred}, \texttt{technicianFor}, \texttt{AcmeInc})$$

An important part of RDF is naming and consistency; for example, the character-string `Fred` is clearly not an ideal Web-scope identifier. User-defined resources (and by extension, classes, properties and datatypes) are thus optionally named using a URI; unnamed resources are represented as *blank-nodes*.[4] (Henceforth, we use Compact URI (CURIE) names [Birbeck and McCarron, 2009] of the form `prefix:reference` to denote URIs, as common in many RDF syntaxes; for example, given a prefix `ex:` which provides a shortcut for the

[3] Arguably, the SPARQL standard for querying RDF contains features which appear to have a Closed World Assumption (e.g., negation-as-failure is expressible using a combination of `OPTIONAL` and `!BOUND` SPARQL clauses) and a Unique Name Assumption (e.g., equals comparisons in `FILTER` expressions). The effects of the Open World Assumption and the lack of a Unique Name Assumption are most overt in OWL.

[4] A certain reading of RDF could view literals as resources, in which case they "identify"—on the level of RDF—their own syntactic form, including the optional language tag and datatype [Klyne and Carroll, 2004]. With inclusion of some datatype-entailment regime, they "identify" some datatype *value*.

URI `http://example.com/ns/`, then `ex:Fred` denotes `http://example.com/ns/Fred`. Note that we give a full list of prefixes used throughout this thesis in Appendix A.)

If required, language tags are denoted by simple strings and *should* be defined in accordance with RFC 3066 [Alvestrand, 2001]. Optional datatypes are inherited from the existing XML Schema standard [Biron and Malhotra, 2004], which defines a set of hierarchical datatypes, associated syntaxes and mapping from the lexical space (i.e., string syntax) to the value space (i.e., interpretable value); RDF defines one additional datatype (`rdf:XMLLiteral`), which indicates a well-formed XML literal.

Additionally, the RDF standard provides a core set of terms useful for describing resources, which we now briefly introduce.

The most prominent RDF term is `rdf:type`, used for stating that a resource is a member of a given class:

(`ex:Fred`, `rdf:type`, `ex:Employee`)

(This is the same as the previous example, but using illustrative CURIEs as names.)

RDF also defines a (meta-)class `rdf:Property` as the class of all properties:

(`ex:age`, `rdf:type`, `rdf:Property`)

Next, RDF defines a set of *containers* which represent groups of things with informally defined semantics [Manola et al., 2004]; viz., `rdf:Bag` denotes the class of unordered containers, `rdf:Seq` denotes the class of ordered containers, and `rdf:Alt` denotes the class of containers denoting alternatives for some purposes. Members of RDF containers are specified using properties of the form `rdf:_`*n*, where for example `rdf:_5` is used to denote the fifth member of the container. However, in practice, RDF containers are not widely used and have been suggested as candidates for deprecation [Berners-Lee, 2010; Feigenbaum, 2010].

Along similar lines, RDF defines syntax for specifying *collections* in the form of a linked list type structure: the collection comprises of elements with a member (a value for `rdf:first`) and a pointer to the next element (a value for `rdf:rest`). As opposed to containers, collections can be closed (using the value `rdf:nil` for `rdf:rest` to terminate the list), such that the members contained within a "well-formed" collection can be interpreted as all of the possible members in that collection. The ability to close the collection has made it useful for standards built on top of RDF, as will be discussed later.

Next, RDF allows for *reificiation*: identifying and describing the RDF triples themselves. One could consider many use-cases whereby such reification is useful (see [Lopes et al., 2010b]); for example, one could *annotate* a triple with its source or *provenance*, an expiration time or other *temporal information*, a *spatial context* within which it holds true, *policies or access rights* for the triple, etc. However, the structure of triples required to perform reification are quite obtuse, where identification and reference to the reified triple requires the following type of construct:

(`ex:FredsAgeTriple`, `rdf:type`, `rdf:Statement`)
(`ex:FredsAgeTriple`, `rdf:subject`, `ex:Fred`)
(`ex:FredsAgeTriple`, `rdf:predicate`, `ex:age`)
(`ex:FredsAgeTriple`, `rdf:object`, `"56"^^xsd:int`)
(`ex:FredsAgeTriple`, `ex:expires`, `"2010-10-28"^^xsd:date`)

Here, the first triple states that the resource `ex:FredsAgeTriple` signifies a triple (aka. a statement), `rdf:subject`, `rdf:predicate` and `rdf:object` are used to reconstruct and thus identify the signified triple, and thereafter arbitrary information can be described, such as the above expiration date. This indirect, verbose modelling has made RDF reification unpopular, and again there have been calls for this usage to be deprecated [Berners-Lee, 2010; Feigenbaum, 2010]. Note also that there is no (or at least, very little) formal semantics defined for the above reification mechanisms.

Finally, RDF supports an explicit means of modelling higher-arity relations using `rdf:value`; for example:

(`ex:Fred`, `ex:technicianFor`, `_:bnode1`)
(`_:bnode1`, `rdf:value`, `ex:AcmeInc`)
(`_:bnode1`, `ex:since`, `"2005-10-28"^^xsd:date`)
(`_:bnode1`, `ex:fulltime`, `"true"^^xsd:boolean`)

Note that `_:bnode1` denotes a blank-node—an anonymous resource—whereby the '_' prefix is reserved in many RDF syntaxes to denote such blank-nodes. Here, `rdf:value` denotes the "primary value" of the `ex:technicianFor` property, with the rest of the triples giving auxiliary information about the relationship.

Finally, various RDF syntaxes have been defined down through the years. The most prominent such syntax is RDF/XML [Beckett and McBride, 2004], which is based on the widely deployed Extensible Markup Language (XML) standard. However, this syntax has been the subject of much criticism. As Beckett—an editor on the (revised) RDF/XML standard—put it:

> *The problems of writing down a graph in a sequential document representing a tree such as the XML DOM has proved just too hard to make it easy to do and clear.* [Beckett, 2010]

However, the RDF/XML syntax has been promoted hand-in-hand with the RDF model itself; RDF/XML still features heavily, for example, in the RDF Primer [Manola et al., 2004].[5] Thus, RDF/XML is widely deployed relative to other RDF syntaxes.

There have been proposals of other RDF syntaxes, each presenting its own unique strengths and weaknesses. Turtle aims to allow "RDF graphs to be completely written in a compact and natural text form, with abbreviations for common usage patterns and datatypes" [Beckett and Berners-Lee, 2008]. N-Triples is a line-delimited (triple-per-line) syntax [Grant and Beckett, 2004, § 3] amenable to simple parsing techniques, line-based processing, and streaming. RDFa—one of the newest syntaxes—allows for embedding RDF data directly into XHTML 1.1 (and, informally, into HTML 5) documents, thus allowing to "augment visual data with machine-readable hints" [Adida and Birbeck, 2008].

In the RDF Semantics W3C Recommendation [Hayes, 2004], RDF is given a model-theoretic—albeit relatively inexpressive—semantics, which gives data described in RDF*meaning*, by means of an *interpretation*. Firstly, these semantics interpret blank-nodes as existential variables—as denoting some unnamed thing known to exist. Next, the *simple interpretation* is defined between graphs; a graph is simply a set of triples, and given one graph known to be true, simple entailment can determine whether a second graph is *necessarily* true or not according to the RDF semantics. Thus, for example, a graph always entails all of its sub-graphs [Hayes, 2004]. Entailment is complicated by the existential nature of blank-nodes, where, for example the singleton graph:

(`_:bnode1`, `ex:worksFor`, `_:bnode2`)

which can be read as "some resource works for some other resource" is entailed by the singleton graph:

(`_:bnodeA`, `ex:worksFor`, `ex:AcmeInc`)

which can be read as "some resource works for Acme Inc.". Thus, (not so) simple entailment checking involves analysis of blank-node rewriting, which has been shown to be NP-complete [Hayes, 2004]. Thereafter, two simple *inference rules* are defined:

[5] We believe this to have caused much confusion amongst RDF novices who (i) conflate the model and the syntax; (ii) do not grasp the triple-centric nature of the RDF model due to obfuscation thereof by the unintuitive RDF/XML syntax.

$$(\texttt{?uuu}, \texttt{?aaa}, \texttt{\_:nnn}) \leftarrow (\texttt{?uuu}, \texttt{?aaa}, \texttt{?xxx})$$
$$(\texttt{\_:nnn}, \texttt{?aaa}, \texttt{?aaa}) \leftarrow (\texttt{?uuu}, \texttt{?aaa}, \texttt{?xxx})$$

Variables—which we prefix with `?`—can match any RDF URI, blank-node, or literal as appropriate. The above two rules state that any RDF triple *simple*-entails the same triple with the subject and/or object replaced with a unique blank-node (`_:nnn`), thus axiomatising the aforementioned semantics of blank-nodes using rules.

The RDF semantics also defines RDF interpretations, and thereafter, RDF entailment. RDF interpretations codify the meaning of triples—in particular, the semantics of properties used in triples—and the meaning of the `rdf:XMLLiteral` datatype. Thereafter, RDF entailment extends upon simple entailment with two additional inference rules: one which states that a resource in the predicate position of triple is a member of the class `rdf:Property`, and another which handles membership of the `rdf:XMLLiteral` datatype. RDF entailment further defines a set of RDF *axiomatic triples*—involving core RDF terms—which always hold true under RDF entailment; for example:

$$(\texttt{rdf:type}, \texttt{rdf:type}, \texttt{rdf:Property})$$

In fact, this set contains countably infinite axiomatic triples of the following form:

$$(\texttt{rdf:\_}n, \texttt{rdf:type}, \texttt{rdf:Property})$$

where n is any natural number. Thus, an empty graph RDF-entails infinite triples.

Note that we include the full set of simple entailment rules, RDF axiomatic triples and RDF entailment rules in § B.2 for reference.

2.2.2 RDF Schema

In April 1998, the first draft of the RDF Schema (RDFS) specification was published as a W3C Working Note [Brickley et al., 1998]. The core idea was to extend upon the relative (semantic) inexpressiveness of RDF and allow for attaching semantics to user-defined classes and properties; the original proposal was to be heavily modified in later versions—for example, features relating to constraints were dropped in favour of a specification more explicitly in tune with the Open World Assumption.

The modern RDFS specification became a W3C Recommendation in early 2004 [Brickley and Guha, 2004]. In particular, RDFS extends RDF with four key terms [Muñoz et al., 2009] which allow for specifying well-defined relationships between classes and properties; viz., `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain` and `rdfs:range`.

Subclassing allows to state that the extension of one class is contained with another:

$$(\texttt{ex:Employee}, \texttt{rdfs:subClassOf}, \texttt{ex:Person})$$

This succinctly states that any member of `ex:Employee` is also a member of `ex:Person`.

Subproperties analogously allow for stating that, e.g.:

$$(\texttt{ex:technicianFor}, \texttt{rdfs:subPropertyOf}, \texttt{ex:worksFor})$$

This states that any two things related by the `ex:technicianFor` property are also related by the `ex:worksFor` property.

Domain and range respectively allow for stating that a resource with any value for a given property is a member of a given class, or that a value for a given property is a member of a given class:

(`ex:technicianFor`, `rdfs:domain`, `ex:Employee`)
(`ex:technicianFor`, `rdfs:range`, `ex:Organisation`)

RDFS also defines some other terms which:

- denote (i) the class of all resources (`rdfs:Resource`), (ii) the class of classes (`rdfs:Class`), (iii) the class of all literals (`rdfs:Literal`), and (iv) the class of all datatypes (`rdfs:Datatype`);
- extend upon the RDF container vocabulary, viz., (i) `rdfs:Container` denoting the super-class of the three aforementioned RDF container classes, (ii) `rdfs:ContainerMembershipProperty` denoting the class of properties of the form `rdf:_`n, and (iii) `rdfs:member` denoting the super-property of all such container membership properties.
- allow for annotating resources with (i) human-readable labels (using the `rdfs:label` property), and (ii) human-readable comments (using the `rdfs:comment` property);
- allow for defining links from one resource to another resource—typically a document—which (i) defines it (using the `rdfs:isDefinedBy` property), or (ii) may contribute relevant information (using the `rdfs:seeAlso` property).

Subsequently, the RDF Semantics recommendation [Hayes, 2004] extends simple/RDF entailment with RDFS entailment, defining a model-theoretic semantics incorporating RDFS terms and providing a set of entailment rules which, by the RDFS entailment lemma [Hayes, 2004, § 7.2], give all possible RDFS entailments for a given graph which do not contain an "XML clash" (an ill-formed `rdf:XMLLiteral` value which represents an inconsistency).[6]

We take RDFS inference rule rdfs2—used to support the semantics of `rdfs:domain`—as an example:

(`?uuu`, `rdf:type`, `?xxx`) ← (`?aaa`, `rdfs:domain`, `?xxx`) , (`?uuu`, `?aaa`, `?yyy`)

This rule states that the subject resource (`?uuu`) with a given property attached (`?aaa`) is a member of the class (`?xxx`) which is defined as the domain of that property. An example application would look like:

(`ex:Fred`, `rdf:type`, `ex:Employee`) ← (`ex:technicianFor`, `rdfs:domain`, `ex:Employee`) ,
(`ex:Fred`, `ex:technicianFor`, `ex:AcmeInc`)

Here, an RDF graph containing the two triples on the right RDFS-entail the triple on the left. Along these lines, there are fourteen RDFS rules, which include rules for supporting entailments possible through the `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain` and `rdfs:range` terms, and which extend upon the two simple entailment rules and two RDF entailment rules; for reference, we list all of RDFS rules and axiomatic triples in § B.2.

It is worth noting that RDFS makes no restriction on how the RDF(S) vocabulary is used, and thus applies to arbitrary RDF graphs and is backwards-compatible with RDF.

2.2.3 Web Ontology Language

Evolving from earlier proposals for Web ontology languages—such as that of SHOE [Luke et al., 1997; Heflin et al., 1999], DAML [Hendler and McGuinness, 2000], OIL [Fensel et al., 2001] and the subsequent hybrid DAML+OIL [McGuinness et al., 2002; Horrocks et al., 2002]—in 2001 the W3C began working

[6] Note that ter Horst ter Horst [2005b] later revealed a "bug" relating to the RDFS entailment lemma, showing that blank-nodes would need to be (temporarily) allowed in the predicate position. Given the prolix nature of the RDFS standard, Muñoz et al. [2009] propose a simpler semantics which only supports the most commonly adopted features.

on a new ontological language which would extend upon RDFS with more expressive semantics, enabling richer entailment regimes: in 2004, the resulting Web Ontology Language (OWL) was recognised as a W3C Recommendation [McGuinness and van Harmelen, 2004].

Like RDFS, OWL can be serialised as RDF (as triples), and abides by the OWA and lack of UNA inherent in RDF. Extending upon RDFS, OWL introduces many new language primitives; a full enumeration is out of scope where we refer the interested reader to [Smith et al., 2004]—for now, we give a brief (incomplete) overview of novel OWL features, and will introduce new primitives in more detail as necessary throughout the rest of the thesis.

Firstly, OWL allows for stating that two resources represent the same thing using `owl:sameAs`:

(`ex:Fred`, `owl:sameAs`, `ex2:Freddy`)

Here, `ex:Fred` and `ex2:Freddy` are stated to be two coreferent identifiers for the same person, which, in particular, allows for aligning the naming schemes used by different sources. Conversely, the property `owl:differentFrom` can be used to explicitly state that two identifiers do *not* refer to the same thing.

Reusing some of the RDFS vocabulary, OWL defines a richer set of primitives for describing classes and properties used in RDF data. For example, OWL allows for defining equivalence between classes, stating that their extension (set of members) is necessarily the same:

(`ex:Human`, `owl:equivalentClass`, `ex:Person`)

More complex class descriptions can be defined based on other class and property terms; for example, classes can be defined as the union, complement or intersection of other classes, as an enumeration of members (e.g., defining the class of `USAStates` by listing precisely all of the states it contains), or in terms of the values of certain properties attached to its members (e.g., defining the class `Parent` as any resource who has a value for the property `hasChild`), etc.

OWL also extends upon the RDFS vocabulary for describing properties; for example, properties can additionally be stated to be inverses of each other:

(`ex:technicianFor`, `owl:inverseOf`, `ex:hasTechnician`)

Properties can also be defined to be equivalent to each other, analogous to class-equivalence exemplified above. OWL also defines four classes of properties which are used to delineate transitive properties, symmetric properties, functional properties (properties whose subject uniquely "identifies" a given value), and inverse-functional properties (properties whose value uniquely "identifies" a given subject).

Additionally, OWL provides an extended set of terms for publishing descriptive metadata, including `owl:versionInfo`, `owl:priorVersion`, `owl:backwardCompatibleWith`, `owl:incompatibleWith`, `owl:DeprecatedClass`, `owl:DeprecatedProperty` primitives used for simple versioning, etc.

On top of all of these primitives—and those inherited from RDF(S)—OWL defines three sublanguages, with different levels of expressivity and different computational properties; viz., OWL Full, OWL DL and OWL Lite.

The most expressive language is OWL Full, which makes no restriction on how the OWL language is used—thus, OWL Full is backwards-compatible with RDFS. However, all typical reasoning tasks over an OWL Full ontology—such as consistency checking, satisfiability checking (checking if a class can consistently have a non-empty extension), checking if one class subsumes another (checking if a class is necessarily a sub-class of another), instance checking (checking if a resource is a member of a class) and conjunctive query answering (posing complex queries against the ontology and its entailments)—are undecidable.

Thus, OWL also defines two "decidable" variations: OWL DL and OWL Lite. Both of these sublanguages are based on Description Logics (DL) [Baader et al., 2002], which provides formal results on the decidability and complexity of certain reasoning tasks over an ontology using various combinations of formally-defined language primitives. Such reasoning tasks are typically implemented using specialised tableau-based approaches inherited from first-order predicate logic [Schmidt-Schauß and Smolka, 1991]. Thus, by applying certain restrictions on the OWL language informed by research on Description Logics, OWL DL and OWL Lite boast certain computational guarantees not possible for OWL Full, in particular enabling the provision of decidability guarantees for certain reasoning tasks—such guarantees on a syntactic subset of OWL are particularly pertinent for "critical applications" where reasoning tasks must always provide complete *and* correct results.

OWL DL—a syntactic subset of OWL Full—is the more expressive of the pair, and allows restricted use of all OWL primitives. Interestingly, OWL DL is only known to be decidable for consistency, satisfiability and instance checking tasks: the decidability of conjunctive query answering is an open question [Grau et al., 2009].[7] Further, the complexity of the former reasoning tasks is NExpTime-complete with respect to the number of axioms[8] and is open (at least NP-Hard) with respect to the number of triples—thus, despite guarantees of decidability, OWL DL reasoning is intractable for certain (OWL DL valid) inputs, meaning that a decision may not be reached in "acceptable time".

Thereafter, OWL Lite—a syntactic subset of OWL DL—offers a still more lightweight variant of OWL where "reasoners for OWL Lite will have desirable computational properties" [Smith et al., 2004]. However, the complexity of OWL Lite is still ExpTime-complete for consistency, satisfiability and instance checking tasks; unlike OWL DL, it is known to be decidable for conjunctive query answering, but is 2ExpTime-complete with respect to query complexity [Grau, 2007]. Thus, again, despite guarantees of decidability, OWL Lite reasoning is intractable for certain valid inputs.

Addressing these issues (amongst others [Grau et al., 2008]), OWL 2 became a recommendation in 2009 [Hitzler et al., 2009], and introduced new language primitives, new semantics for OWL 2 Full (RDF-Based semantics [Schneider, 2009]) and OWL 2 DL (Direct Semantics based on DL formalisms [Motik et al., 2009b]), and three new *profiles* [Grau et al., 2009]—viz., OWL 2 EL, OWL 2 QL, and OWL 2 RL—which are syntactic subsets of OWL 2 DL targeted at different scenarios.

Novel OWL 2 features include additional primitives for describing classes—such as the ability to define classes with reflexivity for a given property (`owl:hasSelf`), or classes which are the union of multiple pair-wise disjoint classes (`owl:disjointUnionOf`), etc.—and properties—such as collections of properties whose values taken together are unique to a resource (`owl:hasKey`), or relations which are entailed by a "chain" of other relations (`owl:propertyChainAxiom`), as well as new classes for delineating reflexive, irreflexive and asymmetric properties, etc.

Reasoning tasks with respect to OWL 2 Full are again undecidable, and with respect to OWL 2 DL are 2NExpTime-complete with the exception of conjunctive query-answering whose decidability/complexity is still open [Grau et al., 2009].[7] However, the new profiles offer better computational characteristics, with each targeting specific scenarios.

OWL 2 EL is based on the Direct Semantics and is designed primarily for classification tasks (subsumption/instance checking) and allows for expressive class axioms, but disallows certain property axioms: OWL 2 EL is PTime-complete (deterministic polynomial complexity) for all reasoning tasks except query-answering, for which it is PSpace-complete [Grau et al., 2009].

OWL 2 QL is also based on the Direct Semantics and aims to provide *relatively* efficient, sound and complete query-answering, in particular aimed at "query-rewriting" implementations over relational database

[7] Glimm and Rudolph [2010] have proven decidability for conjunctive query entailment with respect to the Description Logic underlying OWL DL, but under the assumption that transitive properties (or properties that entail transitive properties) do not appear as predicates in the query. (They believe that the result extends to OWL 2 DL; they do not currently address a complexity bound.)

[8] An axiom represents an "atomic declaration" in OWL, which may consist of multiple triples.

systems, such that structured queries are expanded to request asserted data that may entail some (sub-)goal of the query. Reasoning tasks are NLogSpace-complete with the exception of query answering which is NP-complete; note that query-answering is tractable (NLogSpace/AC^0) with respect to data complexity, but NP-complete with respect to query complexity [Grau et al., 2009] (note this is the same complexity as for simple entailment). To make these gains in complexity, OWL 2 QL makes heavy restrictions on the use of OWL 2, ending up with expressiveness in the intersection of RDFS and OWL 2 DL [Grau et al., 2009].

OWL 2 RL is designed to be implementable using rule-based *and* tableau-based technologies; in particular, it is based on previous proposals to partially support OWL semantics using rules, such as Description Logic Programs (DLP) proposed by Grosof et al. [2004] and pD* proposed by ter Horst [2005b]. Along these lines, OWL 2 RL is a syntactic subset of OWL 2 with an accompanying set of OWL 2 RL/RDF entailment rules such that the entailments possible for OWL 2 RL though Direct Semantics (typically implemented using tableau-based approaches) are aligned with the entailments given by the OWL 2 RL/RDF entailment rules; the result is a partial axiomatisation of OWL 2 RDF-Based semantics in the form of the OWL 2 RL/RDF entailment rules, compatible with RDF Semantics, and computable using rule engines. OWL 2 RL is PTime-complete for all reasoning tasks, except for query answering which remains NP-complete [Grau et al., 2009]; note that query answering is PTime-complete with respect to data complexity, but (like OWL 2 QL) NP-complete with respect to query-complexity.

2.3 RDF Web Publishing and Linked Data

Early efforts involving RDF(S)/OWL publishing on the Web produced large, insular "'data silos", often a dump of potentially huge RDF documents; such silos included OpenCyc comprising of axioms specifying general knowledge[9], the GALEN ontology for describing medical terminology[10], exports from UniProt describing protein sequences[11], and exports from the WordNet lexical database[12]. Typically, these RDF dumps were/are made available as compressed archives linked from some HTML webpage, or as monolithic RDF documents; a person wishing to consume the data would have to visit the site, browse the webpage, follow the links, download the dump and possibly decompress/pre-process the dump. As such, these dumps contained little or no interlinkage.

One notable exception to the emerging RDF silos was the publishing centred around the Friend Of A Friend (FOAF) community. In early 2000, Brickley and Miller started the "RDF Web Ring" project (or simply "RDFWeb") which was eventually rebranded as FOAF.[13] FOAF provides a lightweight vocabulary containing various terms for describing people; initially comprising of a set of RDF properties, this vocabulary evolved throughout the years to include RDFS and OWL descriptions of properties *and* classes, continuously adapting to community feedback and requirements. Various tools solidified adoption of the FOAF vocabulary; one pertinent example is the simple FOAF-a-Matic generator which was released in 2004, and allows users to fill some personal details into a form, which are then encoded as a FOAF profile (an RDF description of the person using FOAF terms) which the user can download and copy to their Web-space—most interestingly, the generator allows for adding information about friends, and for linking to friends' FOAF profiles, thus creating a web of FOAF data. Thus, FOAF profiles became popular amongst Semantic Web enthusiasts and academics, with more adventurous adopters creating ad-hoc vocabularies to extend the personal details contained within. In early 2004, the LiveJournal site—a blogging platform hosting millions of users—began

[9] `http://www.cyc.com/2004/06/04/cyc`; retr. 2010/11/01

[10] `http://www.co-ode.org/galen/full-galen.owl`; retr. 2010/11/01

[11] `http://www.uniprot.org/`; retr. 2010/11/01

[12] `http://www.w3.org/2006/03/wn/wn20/`; retr. 2010/11/01

[13] `http://www.foaf-project.org/original-intro`; retr. 2010/11/02

exporting FOAF data by default for all users[14], which was followed by analogous exports from other social platforms. Unlike earlier data silos, sites such as LiveJournal publish an individual RDF/XML document for each user, thus enabling low-volume/targeted consumption of the structured data. However, early FOAF data—and RDF in general—exhibited sparse use of URIs to identify people [Hogan et al., 2007a]. Thus, consistent URI naming across documents was not even attempted—instead, common practice was to use inverse-functional properties such as emails and weblog URLs to identify persons [Hogan et al., 2007a]—and there was no *direct* means of finding information about a given resource.

As more and more RDF data were published on the Web and more and more vocabularies became available, there was an eminent need in the community for a set of best practices. The first notable step in this direction was the publication in March 2006 of a W3C Working Note entitled "Best Practice Recipes for Publishing RDF Vocabularies" [Miles et al., 2006], which described URI naming schemes for vocabulary terms, and the HTTP mechanisms that should be used to return information upon lookup of those URIs—these best practices aligned with the then recent Web Architecture W3C Recommendation [Jacobs and Walsh, 2004].

In July 2006, Berners-Lee [2006] published the initial W3C Design Issues document outlining Linked Data principles, rationale and some examples—this generalised the earlier best-practices for vocabularies, similarly espousing use of deferenceable HTTP URIs for naming, and additionally encouraging inclusion of external URIs as a simple form of linking. This in turn spawned some initial efforts to promote and support these principles, with tools such as the Tabulator browser [Berners-Lee et al., 2006] enabling browsing over RDF published on the Web as Linked Data.

The first major boon to Linked Data and associated best-practices came in March 2007, when the W3C Semantic Web Education and Outreach (SWEO) Interest Group announced a new Community Project called "Interlinking Open Data"[15]—subsequently shortened to "Linking Open Data" (LOD)—which was inspired by the growth in *Open Data* published on the Web under liberal licences. The goal of the Linked Open Data project is twofold: (i) to bootstrap the Semantic *Web* by creating, publishing and interlinking RDF exports from these open datasets, and in so doing, (ii) introduce the benefits of RDF and Semantic Web technologies to the broader Open Data community [Bizer et al., 2009a]. The LOD project initially found traction through the efforts of academics and developers in research labs converting existing data—most prominently, the DBpedia project [Bizer et al., 2009b] extracting structured data from the collaboratively edited WIKIPEDIA site—but spread to mainstream corporate entities such as the BBC[16], Thompson Reuters[17], the New York Times[18], and various governmental agencies, resulting in a burgeoning, heterogeneous Web of Data built using Semantic Web standards, and augmented with Linked Data principles [Bizer et al., 2009a].

The core message of the Linked Data community is, in essence, a bottom-up approach to bootstrapping Semantic Web publishing. This bottom-up philosophy is best epitomised by the Linked Data "5 Star Scheme" [Berners-Lee, 2006] which was recently added to the Linked Data Design Issues document by Berners-Lee; the scheme is summarised as follows:

⋆ PUBLISH DATA ON THE WEB UNDER AN OPEN LICENSE
⋆ ⋆ PUBLISH STRUCTURED DATA
⋆ ⋆ ⋆ USE NON-PRIORIETARY FORMATS
⋆ ⋆ ⋆ ⋆ USE URIS TO IDENTIFY THINGS
⋆ ⋆ ⋆ ⋆ ⋆ LINK YOUR DATA TO OTHER DATA

—paraphrased from Berners-Lee [2006]

[14] `http://community.livejournal.com/ljfoaf/`; retr. 2010/11/02

[15] See `http://www.w3.org/2005/06/blog/SWEOBlog.php?blog=14&posts=8&page=1&paged=2`; retr. 2010/11/01.

[16] See `http://www.bbc.co.uk/blogs/bbcinternet/2010/02/case_study_use_of_semantic_web.html`; retr. 2011/02/21

[17] `http://www.opencalais.com/`; retr. 2011/02/21

[18] `http://data.nytimes.com/`; retr. 2011/02/21

Here, each additional star is promoted as increasing the potential reusability and interoperability of the publishers' data.[19]

2.4 RDF Search Engines

As RDF publishing on the Web grew in popularity, various applications exploiting this novel source of structured data began to emerge: these included new RDF search engines/warehouses (or more modernly, Linked Data search engines/warehouses) which locate, retrieve, process, index and provide search and querying over RDF data typically gathered from a large number of Web sources. Such search engines may serve a variety of purposes centring around locating pertinent sources of structured information about a given topic or resource, displaying all known information about a given artefact (resource, document, etc.), or answering structured queries posed by users (or user-agents).

Early discussion of structured Web data search engines was provided by Heflin et al. [1999]; whilst discussing potential applications of their proposed SHOE language for annotating webpages, the authors detail requirements for a query-engine with inference support, information gathering through crawling, and subsequent information processing. A competing proposal at the time was Ontobroker [Decker et al., 1998], which also introduced a language for annotating webpages with structured information, and which proposed a mature warehouse architecture including a crawler and extraction component for building a corpus from suitable HTML annotations, an inference engine, a query interface for posing structured queries against the corpus, and an API exposing RDF. As opposed to SHOE, Ontobroker proposed a closed set of ontologies agreed upon by the warehouse which supports them and the data providers that instantiate them. Note that works on the above two systems were concurrent with the initial development of RDF and RDFS and completely predated OWL; as Semantic Web standards matured, so too did the warehouses indexing Semantic Web data.

The earliest "modern" Semantic Web warehouse—indexing RDF(S) and OWL data—was Swoogle [Ding et al., 2004].[20] Swoogle offers search over RDF documents by means of an inverted keyword index and a relational database [Ding et al., 2004]. Given a user-input keyword query, Swoogle will return ontologies, assertional documents and/or terms which mention that keyword (in an RDF literal), thus allowing for discovery of structured information using primitives familiar to Web users from engines such as Google; Swoogle also offers access to software agents [Ding et al., 2004]. To offer such services, Swoogle uses the Google search engine to find documents with appropriate file extensions indicating RDF data, subsequently crawls outwardly from these documents, ranks retrieved documents using links-analysis techniques (inspired by the PageRank algorithm [Page et al., 1998] used by Google), and indexes documents using an inverted keyword index and similarity measures inspired by standard Information Retrieval engines (again, for example, as used by Google [Brin and Page, 1998]). As such, Swoogle leverages traditional document-centric techniques for indexing RDF(S)/OWL documents, with the addition of term search.

Along these lines, we later proposed the Semantic Web Search Engine [Harth and Gassert, 2005; Hogan et al., 2007b; Harth, 2010; Hogan et al., 2010b][21] as a domain-agnostic means of searching for information about resources themselves, as opposed to offering links to related documents: we call this type of search *entity-centric* where an entity is a resource whose description has specifically been amalgamated from numerous (possibly) independent sources.[22] Thus, the unit of search moves away from documents and towards

[19] Please see `http://lab.linkeddata.deri.ie/2010/star-scheme-by-example/` (retr. 2011/01/22) for the rationale behind these stars. Note that although the final star does not explicitly mention Linked Data or RDF, use of these technologies is implied.

[20] System available at `http://swoogle.umbc.edu/`; retr. 2010/10/03.

[21] System available at `http://swse.deri.org/`; retr. 2011/03/01

[22] Note that we did not coin the term "entity-centric", but nor can we pinpoint its precise origins or etymology.

entities which are referential to (possibly) real-world things. Along these lines, we proposed scalable methods for (i) crawling structured data from the Web [Harth et al., 2006], (ii) determining which resources correspond to the same entity, and thereafter consolidating the data by means of identifier canonicalisation [Hogan et al., 2007a]; (iii) performing reasoning to discover new information about entities [Hogan et al., 2009b, 2010c]; (iv) indexing and querying these structured data using the standardised query language SPARQL [Harth et al., 2007]; (v) ranking entities and results [Hogan et al., 2006; Harth et al., 2009]; and (vi) offering user search over the enhanced structured corpus [Hogan et al., 2007b; Harth, 2009]. As RDF Web publishing matured—and with the advent of Linked Data principles—we have adapted our architecture and algorithms accordingly; a recent summary of SWSE research is presented in [Hogan et al., 2010b] and a search prototype is available at `http://swse.deri.org/`. The SWSE system offers much of the motivation behind this thesis, where we particularly focus on points (ii) and (iii) above.

In parallel to the development of SWSE, researchers working on the Falcons Search engine[23] had similar goals in mind: offering entity-centric searching for entities (and concepts) over RDF data sourced from the Web [Cheng et al., 2008a; Cheng and Qu, 2009]. Evolving from the Falcon-AO ontology matching service, the Falcons service operates over arbitrary RDF Web data and also contains components for crawling, parsing, organising, ranking, storing and querying structured data. Like us, they include reasoning, but focus on class-based inferencing—namely class inclusion and instance checking—where class hierarchies and memberships are used to quickly restrict initial results [Cheng and Qu, 2009]. More recently, the authors have proposed a means of identifying coreferent resources (referring to the same entity) based on the semantics of OWL [Hu et al., 2010] and various heuristics.

WATSON also provides keyword search facilities over Semantic Web documents and over entities,[24] but mainly focuses on providing an API to expose services to interested software agents: these services currently include keyword search over indexed RDF documents, retrieving metadata about documents, searching for documents mentioning a given entity, searching for entities matching a given keyword in a document, retrieving class hierarchy information, retrieving entity labels and retrieving triples where a given entity appears in the subject or object position [Sabou et al., 2007; d'Aquin et al., 2007, 2008].

Developed in parallel, Sindice[25] offers similar services to WATSON, originally focussing on providing an API for finding documents which reference a given RDF entity [Oren and Tummarello, 2007], soon extending to keyword-search functionality [Tummarello et al., 2007], inclusion of consolidation using inverse-functional properties [Oren et al., 2008], "per-document" reasoning [Delbru et al., 2008], and simple structured queries [Delbru et al., 2010a]. As such, Sindice have adopted a bottom-up approach, incrementally adding more and more services as feasible and/or required. A more recent addition is that of entity search in the form of Sig.ma[26], which accepts a user-keyword query and returns a description of the primary entity matching that entity as collated from numerous diverse sources [Tummarello et al., 2009] (this can be an appealingly simple form of search, but one which currently assumes that there is only one possible entity of interest for the input query).

As such, there are a number of systems which offer search and/or browsing over large heterogeneous corpora of RDF sourced from the Web, hoping to exploit the emergent Web of Data, where this thesis is inspired by works on SWSE, but where the results apply to any such system, particularly (but not restricted to) systems which offer entity-centric search.

[23] System available at `http://iws.seu.edu.cn/services/falcons/documentsearch/`; *retr.* 2011/01/23

[24] System available at `http://watson.kmi.open.ac.uk/WatsonWUI/`; retr. 2010/11/02

[25] System available at `http://sindice.com/`; retr. 2010/11/02

[26] System available at `http://sig.ma`; retr. 2010/11/02

Chapter 3

Notation and Core Concepts

"If I had eight hours to chop down a tree, I'd spend six hours sharpening my ax."

—Abraham Lincoln

Herein, we provide core preliminaries and notation used throughout the rest of the thesis, relating to (i) RDF (§ 3.1); (ii) Turtle syntax (§ 3.2) (iii) Linked Data principles and data sources (§ 3.3); (iv) atoms and rules (§ 3.4); and (v) terminological data given by RDFS/OWL (§ 3.5). We attempt to preserve notation and terminology as prevalent in the literature. Finally, in § 3.6, we describe our abstract distributed architecture for parallelising the execution of tasks over a cluster of commodity hardware.

3.1 RDF

We briefly give some necessary notation relating to RDF constants and RDF triples; see [Hayes, 2004].

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 $\mathsf{C} := \mathsf{U} \cup \mathsf{B} \cup \mathsf{L}$. The set of literals includes plain literals (which may have an associated language tag) and datatype literals. Note that we interpret blank-nodes as skolem constants signifying particular individuals, as opposed to existential variables as prescribed by the RDF Semantics [Hayes, 2004]. Also, we rewrite blank-node labels when merging documents to ensure uniqueness of labels across those documents [Hayes, 2004].

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* [Grau et al., 2009], 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.

3.2 Turtle Syntax

Throughout this thesis, we may use Terse RDF (Turtle) syntax [Beckett and Berners-Lee, 2008] to denote RDF triples, triple patterns and graphs. Firstly, Turtle syntax allows the use of CURIEs [Birbeck and McCarron, 2009] (e.g., `ex:Fred`) as previously introduced to provide abbreviated URIs—again, a full list of prefixes used in this thesis is available for reference in Appendix A. Secondly, the ‘_’ underscore prefix is reserved for denoting blank-nodes. Thirdly, literals are represented between ‘`"`’ double-quotes; language tags are appended onto literals using the ‘`@`’ at-symbol delimiter; assigned datatypes are appended after a ‘`^^`’ double-caret delimiter. Tokens and terms are white-space delimited (no differentiation between spaces, tabs, new-lines and carriage returns), and triples are (typically) delimited by ‘.’ period characters.

To give some examples of the main syntax and shortcuts:

```
_:bnode1 rdfs:comment "Usually, I am existential"^^xsd:string .
         _:bnode1 rdfs:comment "Herein, I simply am"@en .
```

Further, triples which share a given subject can be grouped together using a ‘`;`’ semi-colon delimiter:

```
_:bnode1 rdfs:comment "Usually, I am existential"^^xsd:string ;
              rdfs:comment "Herein, I simply am"@en .
```

Triples which share a common subject and predicate can be grouped together using a ‘`,`’ comma delimiter:

```
_:bnode1 rdfs:comment "Usually, I am existential"^^xsd:string ,
                      "Herein, I simply am"@en .
```

To avoid having to invent arbitrary blank-node labels, ‘`[ ]`’ square brackets can alternately be used:

```
[ rdfs:comment "Usually, I am existential"^^xsd:string ,
               "Herein, I simply am"@en .  ]
```

Syntax is also provided to abbreviate ordered RDF collections (a.k.a. lists) and boolean and numeric datatype literals:

```
ex:Fred ex:top5Values ( ex:Phi 1836.2 -5 false 3e2 ) .
```

where the above syntax would expand to the more verbose:

```
ex:Fred ex:top5Values [  rdf:first ex:Phi ; rdf:rest
                      [  rdf:first "1836.2"^^xsd:decimal ; rdf:rest
                      [  rdf:first "-5"^^xsd:integer ; rdf:rest
                      [  rdf:first "false"^^xsd:boolean ; rdf:rest
                      [  rdf:first "3e2"^^xsd:double ; rdf:rest
                         rdf:nil ] ] ] ] ] .
```

Finally, the simple term ‘`a`’ can be used as a shortcut for `rdf:type`:

```
ex:Fred a foaf:Person .
```

3.3 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.

Linked Data Principles Throughout this thesis, we will refer to the four best practices of Linked Data as follows [Berners-Lee, 2006]:

- (**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 [Fielding et al., 1999] 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 [Hayes, 2004]. 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 [Fielding et al., 1999]; 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)—note that we do not need to distinguish between the different `30x` redirection schemes, and that this function would implicitly involve, e.g., stripping the fragment identifier of a URI [Berners-Lee et al., 2005]. 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.

3.4 Atoms and Rules

In this section, we briefly introduce some notation as familiar particularly from the field of Logic Programming [Lloyd, 1987], which eventually gives us the notion of a *rule*—a core concept for reasoning. 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. (In particular, the Logic Programming formalisms presented herein allow for more clearly bridging to the work of Kifer and Subrahmanian [1992] on annotated logic programs, which will be central to Chapter 6.)

Atom An *atomic formula* or *atom* is a formula of the form $p(e_1, \ldots, e_n)$, where $e_1, \ldots, e_n$ 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. An atom, or its negation, is also known as a *literal*—we henceforth avoid this sense of the term as it is homonymic with an RDF literal. As such, this notation can be thought of as generalising that of RDF triples; note that an *RDF predicate* (the second element of a triple) has its etymology from a predicate such as p above, where triples can be represented as atoms of the form $p(s, o)$—for example, `age(Fred,56)`. However, it is more convenient to consider an RDF predicate as a standard term, and to use a static *ternary predicate* T to represent RDF triples in the form $T(s,p,o)$—for example, T`(Fred, age, 56)`—where we will typically omit T whose presence remains implicit where appropriate.

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 $A \rhd B$—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:

$$H \leftarrow B_1, \ldots, B_n (n \geq 0)\,, \tag{3.1}$$

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.[1] Our rules are *range restricted*, also known as *safe* [Ullman, 1989]: 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.

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:[2]

$$\begin{aligned}\mathfrak{T}_P :\ & 2^{\mathsf{Facts}} \to 2^{\mathsf{Facts}}\\ & I \mapsto \Big\{\mathsf{Head}(R)\theta \mid R \in P \wedge \exists I' \subseteq I \text{ s.t. } \theta = \mathsf{mgu}\big(\mathsf{Body}(R), I'\big)\Big\}\end{aligned}$$

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

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

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

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 $\mathfrak{T}_P$ [Yardeni and Shapiro, 1991] (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 [Ullman, 1989]. 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 $\mathsf{lm}(P)$.

3.5 Terminological Data: RDFS/OWL

As previously described, RDFS/OWL allow for disseminating terminological data—loosely schema-level data—which provide definitions of classes and properties. Herein, we provide some preliminaries relating to our notion of terminological data. (Note that a precise and standard definition of terminological data is somewhat difficult for RDFS and particularly OWL Full; we instead rely on a convenient 'shibboleth' approach which identifies markers for what we consider to be RDFS/OWL 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:Restriction`, `owl:DatatypeProperty`, `owl:FunctionalProperty`, etc.; `rdfs:Resource`, `rdfs:Literal`, e.g., are not 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.; `rdf:type`, `owl:sameAs`, `rdfs:label`, e.g., do *not* have a meta-class as domain.

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.);
4. triples which contribute to an all-disjoint-classes or all-disjoint-properties axiom.[3]

Note that the last category of terminological data is only required for special consistency-checking rules called constraints: i.e., rules which check for logical contradictions in the data. For brevity, we leave this

[3] That is, triples with `rdf:type` as predicate and `owl:AllDisjointClasses` or `owl:AllDisjointProperties` as object, triples whose predicate is `owl:members` and whose subject unifies with the previous category of triples, and triples forming a valid RDF list whose head unifies with the object of such an `owl:members` triple.

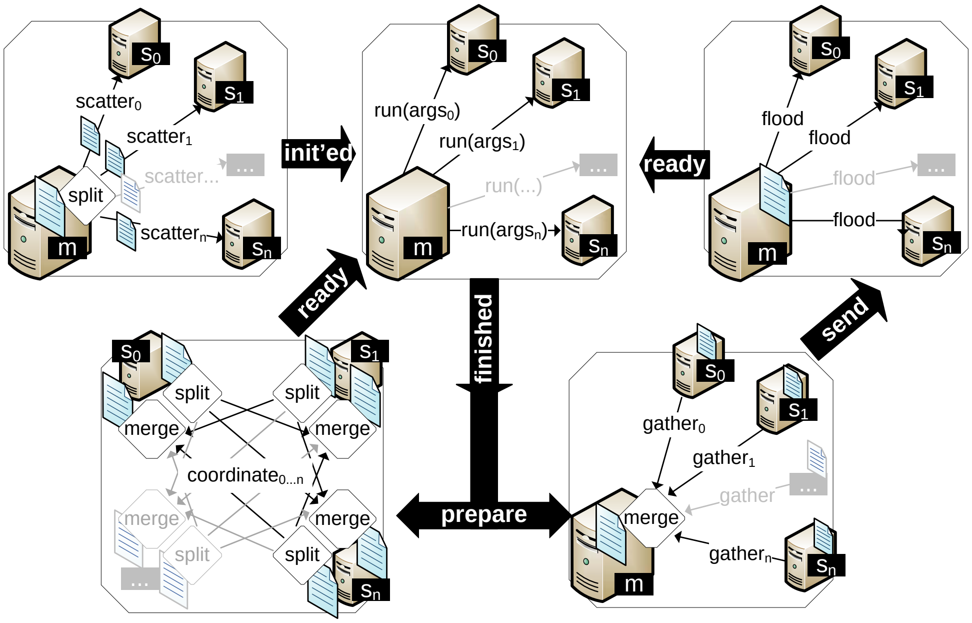


Figure 3.1: Abstract distributed interface

last category of terminological data implicit in the rest of the thesis, where `owl:AllDisjointClasses` and `owl:AllDisjointProperties` can be thought of as "honourary meta-classes" included in category 1, `owl:-members` can be thought of as an "honourary meta-property" included in category 2, and the respective RDF lists included in category 3.

3.6 Distribution Framework

We now introduce the distribution framework upon which all of our methods are implemented.

We employ a shared-nothing distributed architecture [Stonebraker, 1986] over a cluster of commodity hardware. The distributed framework consists of a master machine which orchestrates the given tasks, and several slave machines which perform parts of the task in parallel. The abstract interface of distributed methods is illustrated in Figure 3.1.

The master machine can instigate the following distributed operations:

- **scatter:** partition on-disk data using some local *split* function, and send each chunk to individual slave machines for subsequent processing;

- **run:** request the parallel execution of a task by the slave machines—such a task either involves processing of some data local to the slave machine, or the **coordinate** method (described later) for reorganising the data under analysis;

- **gather:** gathers chunks of output data from the slave swarm and performs some local *merge* function over the data;

- **flood:** broadcast global knowledge required by all slave machines for a future task.

The master machine provides input data to the slave swarm, provides the control logic required by the distributed task (commencing tasks, coordinating timing, ending tasks), gathers and locally perform tasks on global knowledge which the slave machines would otherwise have to replicate in parallel, and transmits globally required knowledge.

The slave machines, as well as performing tasks in parallel, can perform the following distributed operation (on the behest of the master machine):

- **coordinate:** local data on each slave machine are partitioned according to some *split* function, with the chunks sent to individual machines in parallel; each slave machine also gathers the incoming chunks in parallel using some *merge* function.

The above operation allows slave machines to reorganise (split/send/gather) intermediary data amongst themselves; the **coordinate** operation could be replaced by a pair of **gather/scatter** operations performed by the master machine, but we wish to avoid the channelling of all intermediary data through one machine.

For the experiments run in thesis, we instantiate this architecture using the standard Java Remote Method Invocation libraries as a convenient means of development given our Java code-base.

All of our evaluation is based on nine machines connected by Gigabit ethernet[4], 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 thesis 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.

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

Chapter 4

Crawling, Corpus and Ranking[*]

"Be conservative in what you do; be liberal in what you accept from others"

—**Postel's Law**

In this chapter, we provide some initial contributions necessary for the core work presented in Chapters 5–7. In particular, we describe the design and implementation of our distributed Linked Data crawler for retrieving large amounts of RDF data from the Web. Subsequently, we detail the parameters of the crawl we performed to achieve an evaluation corpus of 1.118 billion quadruples from 3.985 million sources, presenting a selection of pertinent statistics from the data—this corpus is used throughout the rest of the thesis. Finally, we present our distributed implementation of an algorithm—introduced by Harth et al. [2009]—for performing links-based ranking of Linked Data documents; we use the results of this ranking procedure in Chapter 5 when analysing the use of RDFS and OWL in Linked Data, and also in Chapter 6 as the basis for annotating triples with ranking scores.

4.1 Crawler

Our crawler is designed to retrieve large amounts (in the order of billions of triples) of RDF data from the Web by following links between documents; we thus draw inspiration from traditional (HTML-centric) Web crawlers, with which we share the following requirements:

- **Politeness:** The crawler must implement politeness restrictions to avoid hammering remote servers with dense HTTP GET requests and to abide by policies identified in the provided `robots.txt`[1] files.
- **Throughput:** The crawler should crawl as many URIs as possible in as little time as is possible within the bounds of the politeness policies.
- **Scale:** The crawler must be able to locate, retrieve and process millions of documents.
- **Quality:** The crawler should prioritise crawling URIs it considers to be "high quality".

Our crawler starts with a set of seed URIs referring to RDF documents, retrieves the content of URIs, parses and writes content to disk in the form of quads, and recursively extracts new URIs for crawling.

[*] Parts of this chapter have been submitted for review as [Hogan et al., 2010b].

[1] See `http://www.robotstxt.org/orig.html`; retr. 2010/12/01

Following **LDP2** and **LDP3** (see § 3.3), we consider all `http:` protocol URIs extracted from an RDF document (as found in either the subject, predicate or object position of a triple) as candidates for crawling. Additionally—and specific to crawling structured data—we identify the following requirement:

- **Structured Data:** The crawler should retrieve a high percentage of RDF/XML documents and avoid wasted lookups on unwanted formats: e.g., HTML documents.

Currently, we crawl for RDF/XML syntax documents where RDF/XML is still the most commonly used syntax for publishing RDF on the Web.[2]

Algorithm 4.1 outlines the operation of the crawler, which will be explained in detail throughout this section.[3] Although we only extract links from RDF/XML, note that all of our methods apply to generic crawling, with the exception of some optimisations for maximising the ratio of RDF/XML documents (discussed in § 4.1.5).

4.1.1 Breadth-first Crawling

Traditional Web crawlers (e.g., see [Heydon and Najork, 1999; Boldi et al., 2002]) typically use a breadth-first crawling strategy: the crawl is conducted in rounds, with each round crawling a *frontier*. On a high-level, Algorithm 4.1 represents this round-based approach applying *ROUNDS* number of rounds. The frontier comprises of seed URIs for round 0 (Line 1, Algorithm 4.1), and thereafter with novel URIs extracted from documents crawled in the previous round (Line 18, Algorithm 4.1). Thus, the crawl emulates a breadth-first traversal of inter-linked Web documents. (Note that the algorithm is further tailored according to requirements we will describe as the section progresses.)

As we will see later in the section, the round-based approach fits well with our distributed framework, allowing for crawlers to work independently for each round, and coordinating new frontier URIs at the end of each round. Additionally, Najork and Wiener [2001] show that a breadth-first traversal strategy tends to discover high-quality pages early on in the crawl, with the justification that well-linked documents (representing higher quality documents) are more likely to be encountered in earlier breadth-first rounds; similarly, breadth first crawling leads to a more diverse dataset earlier on, rather than a depth-first approach which may end up traversing deep paths within a given site. [Lee et al., 2008] justify a rounds-based approach to crawling based on observations that writing/reading concurrently and dynamically to a single queue can become the bottleneck in a large-scale crawler.

4.1.2 Incorporating Politeness

The crawler must be careful not to bite the hands that feed it by hammering the servers of data providers or breaching policies outlined in the provided `robots.txt` file [Thelwall and Stuart, 2006]. We use pay-level-domains [Lee et al., 2008] (PLDs; a.k.a. "root domains"; e.g., `bbc.co.uk`) to identify individual data providers, and implement politeness on a per-PLD basis. Firstly, when we first encounter a URI for a PLD, we cross-check the `robots.txt` file to ensure that we are permitted to crawl that site; secondly, we implement a "minimum PLD delay" to avoid hammering servers, viz.: a minimum time-period between subsequent requests to a given PLD. This is given by *MINDELAY* in Algorithm 4.1—we currently allow two lookups per domain per second.[4]

[2] In future to extend the crawler to support other formats such as RDFa, N-Triples and Turtle—particularly given the increasing popularity of the former syntax.

[3] Algorithm 4.1 omits some details for brevity—e.g., checking `robots.txt` policies.

[4] We note that different domains have different guidelines, and our policy of two lookups per second may be considered conservative for many providers; e.g., see `http://www.livejournal.com/bots/` (retr. 2010/01/10) which allows up to five lookups per second. However, we favour a more conservative policy in this regard.

Algorithm 4.1: Algorithm for crawling

```
Require: $SEEDS$, $ROUNDS$, $PLDLIMIT$, $MINDELAY$
 1: $F \leftarrow \{(u,1) \mid u \in SEEDS\}$        /* frontier with inlink count: $F : \mathsf{U} \rightarrow \mathbb{N}$ */
 2: $Q \leftarrow \emptyset$        /* per-PLD queue: $Q := (P_1, \ldots, P_n), P_i : \mathsf{U} \times \ldots \times \mathsf{U}$ */
 3: $R \leftarrow \emptyset$        /* RDF/non-RDF counts for a pld: $R : \mathsf{U} \rightarrow \mathbb{N} \times \mathbb{N}$ */
 4: $S \leftarrow \emptyset$        /* seen list: $S \subset \mathsf{U}$ */
 5: for $r \leftarrow 1$ to $ROUNDS$ do
 6:   fill($Q, F, S, PLDLIMIT$)        /* add highest linked $u$ to each PLD queue */
 7:   for $d \leftarrow 1$ to $PLDLIMIT$ do
 8:     $start \leftarrow$ current_time()
 9:     for $P_i \in Q$ do
10:       $cur \leftarrow$ calculate_cur($P_i$, $R$)        /* see § 4.1.5 */
11:       if $cur >$ random([0,1]) then
12:         poll $u$ from $P_i$
13:         add $u$ to $S$
14:         $u_{\mathsf{deref}} \leftarrow \mathsf{deref}(u)$
15:         if $u_{\mathsf{deref}} = u$ then
16:           $G \leftarrow \mathsf{get}(u)$
17:           for all $u_G \in$extractHttpURIs($G$) do
18:             $F(u_G)$++        /* $F(u_G) \leftarrow 1$ if novel */
19:           end for
20:           output $G$ to disk
21:           update $R$        /* based on whether $G = \emptyset$ or $G \neq \emptyset$ */
22:         else
23:           $F(u) \rightarrow F(u_{\mathsf{deref}})$        /* add & link counts for $u$ to $u_{\mathsf{deref}}$ */
24:         end if
25:       end if
26:     end for
27:     $elapsed \leftarrow$ current_time() - $start$
28:     if $elapsed < MINDELAY$ then
29:       wait($MINDELAY - elapsed$)
30:     end if
31:   end for
32: end for
```

In order to accommodate the min-delay policy with minimal effect on performance, we must refine our crawling algorithm: large sites with a large internal branching factor (large numbers of unique intra-PLD outlinks per document) can result in the frontier of each round being dominated by URIs from a small selection of PLDs. Thus, naïve breadth-first crawling can lead to crawlers hammering such sites; conversely, given a politeness policy, a crawler may spend a lot of time idle waiting for the min-delay to pass.

One solution is to reasonably restrict the branching factor [Lee et al., 2008]—the maximum number of URIs crawled per PLD per round—which ensures that individual PLDs with large internal fan-out are not hammered; thus, in each round of the crawl, we implement a cut-off for URIs per PLD, given by *PLDLIMIT* in Algorithm 4.1.

Secondly, to enforce the politeness delay between crawling successive URIs for the same PLD, we implement a per-PLD queue (given by Q in Algorithm 4.1) whereby each PLD is given a dedicated queue of URIs filled from the frontier, and during the crawl, a URI is polled from each PLD queue in a round-robin fashion. If all of the PLD queues have been polled before the min-delay is satisfied, then the crawler must wait: this is given by Lines 27-30 in Algorithm 4.1. Thus, the minimum crawl time for a round—assuming a sufficiently full queue—becomes *MINDELAY* * *PLDLIMIT*.

4.1.3 On-disk Queue

As the crawl continues, the in-memory capacity of the machine will eventually be exceeded by the capacity required for storing URIs [Lee et al., 2008].[5] In order to scale beyond the implied main-memory limitations of the crawler, we implement on-disk storage for URIs, with the additional benefit of maintaining a persistent state for the crawl and thus offering a "continuation point" useful for extension of an existing crawl, or recovery from failure.

We implement the on-disk storage of URIs using Berkeley DB [Olson et al., 1999] which comprises of two indexes—the first provides lookups for URI strings against their status (polled/unpolled); the second offers a key-sorted map which can iterate over unpolled URIs in decreasing order of inlink count. The inlink count reflects the total number of documents from which the URI has been extracted thus far; we deem a higher count to roughly equate to a higher priority URI (following similar intuition as links-analysis techniques such as PageRank [Page et al., 1998] whereby we view an inlink as a positive vote for the content of that document).

The crawler utilises both the on-disk index and the in-memory queue to offer similar functionality as above. The on-disk index and in-memory queue are synchronised at the start of each round:

1. links and respective inlink counts extracted from the previous round (or seed URIs if the first round) are added to the on-disk index;

2. URIs polled from the previous round have their status updated on-disk;

3. an in-memory PLD queue—representing the candidate URIs for the round—is filled using an iterator of on-disk URIs sorted by descending inlink count.

Most importantly, the above process ensures that only the URIs active (current PLD queue and frontier URIs) for the current round must be stored in memory. Also, the process ensures that the on-disk index stores the persistent state of the crawler up to the start of the last round; if the crawler (or machine, etc.) unexpectedly fails, the crawl can be resumed from the start of the last round. Finally, the in-memory PLD queue is filled with URIs sorted in order of inlink count, offering a cheap form of intra-PLD URI prioritisation (Line 6, Algorithm 4.1).

4.1.4 Multi-threading

The bottle-neck for a single-threaded crawler will be the response times of remote servers; the CPU load, I/O throughput and network bandwidth of a crawling machine will not be efficiently exploited by sequential HTTP GET requests over the Web. Thus, crawlers are commonly multi-threaded to mitigate this bottleneck and perform concurrent HTTP lookups. At a certain point of increasing the number of active lookup threads, the CPU load, I/O load, *or* network bandwidth becomes an immutable bottleneck with respect to local hardware (not dependant on remote machines).

In order to find a suitable thread count for our particular setup (with respect to processor/network bandwidth), we conducted some illustrative small-scale experiments comparing a machine crawling with the same setup and input parameters, but with an exponentially increasing number of threads: in particular,

[5] By means of illustration, we performed a stress-test and observed that with 2GB of JAVA heap-space, our implementation could crawl approx. 199 thousand URIs (additionally storing the respective frontier URIs) before throwing an out-of-memory exception.

we measure the time taken for crawling 1,000 URIs given a seed URI[6] for 1, 2, 4, 8, 16, 32, 64, and 128 threads.[7]

For the different thread counts, Figure 4.1 overlays the total time taken in minutes to crawl the 1,000 URIs, and also overlays the average percentage CPU *idle* time.[8] Time and CPU% idle noticeably have a direct correlation. As the number of threads increases up until 64, the time taken for the crawl decreases—the reduction in time is particularly pronounced in earlier thread increments; similarly, and as expected, the CPU idle time decreases as a higher density of documents are retrieved and processed. Beyond 64 threads, the effect of increasing threads becomes minimal as the machine reaches the limits of CPU and disk I/O throughput; in fact, the total time taken starts to increase – we suspect that contention between threads for shared resources affects performance. Thus, we settle upon 64 threads as an approximately optimal figure for our setup.

Note that we can achieve a similar performance boost by distributing the crawl over a number of machines; we will see more in § 4.1.6.

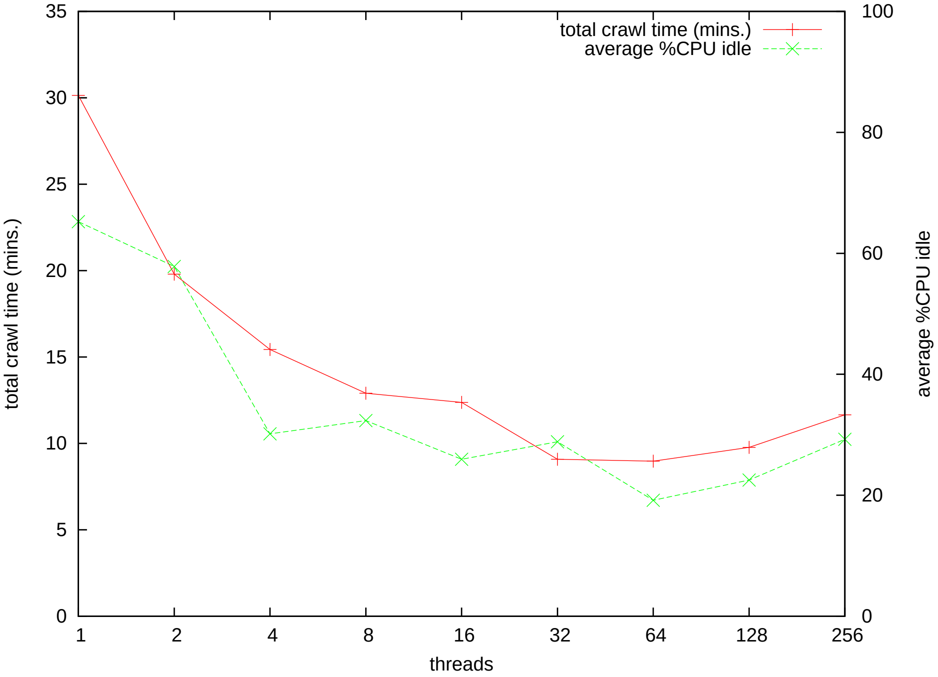


Figure 4.1: Total time (mins.) and average percentage of CPU *idle* time for crawling 1,000 URIs with a varying number of threads

4.1.5 Crawling RDF/XML

Given that we currently only handle RDF/XML documents, we would feasibly like to maximise the ratio of HTTP lookups which result in RDF/XML content; i.e., given the total HTTP lookups as L, and the total number of downloaded RDF/XML pages as R, we would like to maximise the ratio R/L.

In order to reduce the amount of HTTP lookups wasted on non-RDF/XML content, we implement the

[6] `http://aidanhogan.com/foaf/foaf.rdf`

[7] We pre-crawled all of the URIs before running the benchmark to help ensure that the first test was not disadvantaged by a lack of remote caching.

[8] Idle times are measured as (100 - %CPU Usage), where CPU usage is extracted from the UNIX command `ps` taken every three seconds during the crawl.

following heuristics:

1. firstly, we blacklist non-`http(s)` URI schemes (e.g., `mailto:`, `file:`, `fax:`, `ftp:`[9], `tel:`);
2. secondly, we blacklist URIs with common file-extensions that are highly unlikely to return RDF/XML (e.g., `html`, `jpg`, `pdf`, etc.) following arguments we previously laid out in [Umbrich et al., 2008];
3. thirdly, we check the returned HTTP header and only retrieve the content of URIs reporting `Content-type: application/rdf+xml`;[10]
4. finally, we use a *credible useful ratio* when polling PLDs to indicate the probability that a URI from that PLD will yield RDF/XML based on past observations.

Although the first two heuristics are quite trivial and should still offer a high theoretical recall of RDF/XML, the third is arguable in that previous observations [Hogan et al., 2010a] indicate that 17% of RDF/XML documents are returned with a `Content-type` other than `application/rdf+xml`—it is indeed valid (although not considered best practice) to return more generic content-types for RDF/XML (e.g., `text/xml`)—where we automatically exclude such documents from our crawl; however, here we put the onus on publishers to ensure reporting of the most specific `Content-type`.

With respect to the fourth item, we implement an algorithm for selectively polling PLDs based on their *observed useful ratio*: the percentage of documents thus far retrieved from that domain which the crawler deems useful. Since our crawler only requires RDF/XML, we use this score to access PLDs which offer a higher percentage of RDF/XML more often (Lines 10 & 11, Algorithm 4.1). Thus, we can reduce the amount of time wasted on lookups of HTML documents and save the resources of servers for non-RDF/XML data providers.

The credible useful ratio for PLD i is derived from the following credibility formula:

$$cur_i = \frac{rdf_i + \mu}{total_i + \mu}$$

where rdf_i is the total number of RDF documents returned thus far by PLD i, $total_i$ is the total number of lookups performed for PLD i excluding redirects, and μ is a "credibility factor". The purpose of the credibility formula is to dampen scores derived from few readings (where $total_i$ is small) towards the value 1 (offering the benefit-of-the-doubt), with the justification that the credibility of a score with few readings is less than that with a greater number of readings: with a low number of readings ($total_i \ll \mu$), the cur_i score is affected more by μ than actual readings for PLD i; as the number of readings increases ($total_i \gg \mu$), the score is affected more by the observed readings than the μ factor. We set this constant to 10;[11] thus, for example, if we observe that PLD a has returned 1/5 RDF/XML documents and PLD b has returned 1/50 RDF/XML documents, then $cur_a = (1+\mu)/(5+\mu) = 0.7\overline{3}$ and $cur_b = (1+\mu)/(50+\mu) = 0.18\overline{3}$—we thus ensure that PLDs are not unreasonably punished for returning non-RDF/XML documents early on.

To implement selective polling of PLDs according to their useful ratio, we simply use the cur score as a probability of polling a URI from that PLD queue in that round (Lines 10-11, Algorithm 4.1). Thus, PLDs which return a high percentage of RDF/XML documents—or indeed PLDs for which very few URIs have been encountered—will have a higher probability of being polled, guiding the crawler away from PLDs which return a high percentage of non RDF/XML documents.

[9] Admittedly, the `ftp:` scheme may yield valid RDF/XML, but for the moment we omit the scheme.

[10] Indeed, one advantage RDF/XML has over RDFa is an unambiguous MIME-type which is useful in such situations—RDFa is typically served as `application/xhtml+xml`.

[11] Admittedly, a 'magic number'; however, the presence of such a factor is more important than its actual value: without the credibility factor, if the first document returned by a PLD was non-RDF/XML, then that PLD would be completely ignored for the rest of the crawl

PLD	polled	skipped	*ur*	*cur*	% polled
linkedmdb.org	1980	0	1	1	100
geonames.org	1971	26	0.996	0.996	98.7
rdfabout.com	1944	0	1	1	100
fu-berlin.de	1925	74	0.947	0.949	96.3
bbc.co.uk	1917	33	0.982	0.982	98.3
deri.ie	233	1814	0.013	0.054	11.4
megginson.com	196	1795	0	0.049	9.8
xbrl.us	206	1785	0	0.046	10.3
wikipedia.org	203	1784	0	0.051	10.2
uklug.co.uk	215	1776	0	0.044	10.7

Table 4.1: Useful ratio (***ur***) and credible useful ratio (***cur***) for the top five most often polled/skipped PLDs

We evaluated the useful ratio scoring mechanism on a crawl of 100k URIs, with the scoring enabled and disabled. In the first run, with scoring disabled, 22,504 of the lookups resulted in RDF/XML (22.5%), whilst in the second run with scoring enabled, 30,713 lookups resulted in RDF/XML (30.7%). Table 4.1 enumerates the top 5 PLDs which were polled and the top 5 PLDs which were skipped for the crawl with scoring enabled, including the useful ratio (*ur*—the unaltered ratio of useful documents returned to non-redirect lookups) and the credible useful ratio score (*cur*). The top 5 polled PLDs were observed to return a high-percentage of RDF/XML, and the top 5 skipped PLDs were observed to return a low percentage of RDF.

4.1.6 Distributed Approach

We have seen that given a sufficient number of threads, the bottleneck for multi-threaded crawling becomes the CPU and/or I/O capabilities of one machine; thus, by implementing a distributed crawling framework balancing the CPU workload over multiple machines, we expect to increase the throughput of the crawl. We apply the crawling over our distributed framework (§ 3.6) as follows:

1. **scatter**: the master machine scatters a seed list of URIs to the slave machines, using a hash-based *split* function;

2. **run**: each slave machine adds the new URIs to its frontier and performs a round of the crawl, writing the retrieved and parsed content to the local hard-disk, and creating a frontier for the next round;

3. **coordinate**: each slave machine then uses the *split* function to scatter new frontier URIs to its peers.

Steps 2 & 3 are recursively applied until *ROUNDS* has been fulfilled. Note that in Step 2, we adjust the *MINDELAY* for subsequent HTTP lookups to a given PLD value by multiplying the number of machines: herein, we somewhat relax our politeness policy (e.g., no more than 8 lookups every 4 seconds, as opposed to 1 lookup every 0.5 seconds), but deem the heuristic sufficient assuming a relatively small number of machines and/or large number of PLDs.

In order to evaluate the effect of increasing the number of crawling machines within the framework, we performed a crawl performing lookups on 100k URIs on 1, 2, 4 and 8 machines using 64 threads. The results are presented in Table 4.2, showing number of machines, number of minutes taken for the crawl, and also the percentage of times that the in-memory queue had to be delayed in order to abide by our politeness policies. There is a clear increase in the performance of the crawling with respect to increasing number of machines. However, in moving from four machines to eight, the decrease in time is only 11.3%. With 8

#machines	**1**	**2**	**4**	**8**
mins	360	156	71	63
%delay	1.8	10	81.1	94.6

Table 4.2: Time taken for a crawl performing lookups on 100 thousand URIs, and average percentage of time each queue had to enforce a politeness wait, for differing numbers of machines

machines (and indeed, starting with 4 machines), there are not enough active PLDs in the queue to fill the adjusted min-delay of 4 seconds (8*500 ms), and so the queue has a delay hit-rate of 94.6%.

We term this state *PLD starvation*: the slave machines do not have enough unique PLDs to keep them occupied until the *MINDELAY* has been reached. Thus, we must modify somewhat the end-of-round criteria to reasonably improve performance in the distributed case:

- firstly, a crawler can return from a round if the *MINDELAY* is not being filled by the active PLDs in the queue—the intuition here being that new PLDs can be discovered in the *frontier* of the next round;
- secondly, in the case that new PLDs are not found in the *frontier*, we implement a *MINPLDLIMIT* which ensures that slave machines don't *immediately* return from the round;
- finally, in the case that one slave crawler returns from a round due to some stopping criteria, the master machine will request that all other slave machines also end their round such that machines do not remain idle waiting for their peers to return.

The above conditions help to somewhat mitigate the effect of PLD starvation on our distributed crawl; however, given the politeness restriction of 500 ms per PLD, this becomes a hard-limit for performance independent of system architecture and crawling hardware, instead imposed by the nature of the Web of Data itself. As a crawl progresses, active PLDs (PLDs with unique content still to crawl) will become less and less, and the performance of the distributed crawler will approach that of a single-machine crawl. As Linked Data publishing expands and diversifies, and as the number of servers hosting RDF content increases, better performance would be observed for distributed crawling on larger numbers of machines: for the moment, we observe that 8 machines currently approaches the limit of performance given our setup and policies.

4.1.7 Related Work

With respect to crawling, parts of our architecture and some of our design decisions are influenced by work on traditional Web crawlers; e.g., the IRLBot system of Lee et al. [2008] and the distributed crawler of Boldi et al. [2002].

Related work in the area of *focused crawling* can be categorised roughly as follows [Batsakis et al., 2009]:

- *classic focused crawling*: e.g., Chakrabarti et al. [1999] use primary link structure and anchor texts to identify pages about a topic using various text similarity of link analysis algorithms;
- *semantic focused crawling*: is a variation of classical focused crawling but uses conceptual similarity between terms found in ontologies [Ehrig and Maedche, 2003; Dong et al., 2009]
- *learning focused crawling*: Diligenti et al. [2000]; Pant and Srinivasan [2005] use classification algorithms to guide crawlers to relevant Web paths and pages.

However, a fundamental difference between these approaches and ours is that our definition of high quality pages is not based on topic, but instead on the content-type of documents.

With respect to RDF, the Swoogle search engine implements a crawler which extracts links from Google, and further crawls based on various—sometimes domain specific—link extraction techniques [Ding et al., 2004]; like us, they also use file extensions to throw away non-RDF URIs. In later work, Ding and Finin [2006] conducted a crawl of 1.7 million RDF documents resulting in 300 million triples which they then analysed and found that, e.g., terms from the `foaf:`, `rss:` and `dc:` namespaces were particularly popular.

Cheng and Qu [2009] provide a very brief description of the crawler used by the FalconS search engine for obtaining RDF/XML content. Interestingly, they provide statistics identifying a power-law like distribution for the number of documents provided by each pay-level domain, correlating with our discussion of PLD-starvation: few domains provide many documents, translating into fewer and fewer domains actively contributing data as the crawl continues.

In [Sabou et al., 2007], for the purposes of the WATSON engine, the authors use Heritrix[12] to retrieve ontologies using Swoogle, Google and Protégé indexes, and also crawl by interpreting `rdfs:seeAlso` and `owl:imports` as links—they do not exploit the dereferencability of URIs popularised by Linked Data.

Similarly, the Sindice crawler [Tummarello et al., 2007] retrieves content based on a *push* model, crawling documents which pinged some central service such as PingTheSemanticWeb[13]; they also discuss a PLD-level scheduler for ensuring politeness and diversity of data retrieved.

4.1.8 Critical Discussion and Future Directions

From a pragmatic perspective, we would prioritise extension of our crawler to handle arbitrary RDF formats, especially the RDFa format which is growing in popularity. Such an extension may mandate modification of the current mechanisms for ensuring a high percentage of RDF/XML documents: for example, we could no longer blacklist URIs with a `.html` file extension, nor could we rely on the `Content-type` returned by the HTTP header (unlike RDF/XML, RDFa does not have a specific MIME-type). Along these lines, we could perhaps also investigate extraction of structured data from non-RDF sources; these could include Microformats, metadata embedded in documents such as PDFs and images, extraction of HTML meta-information, HTML scraping, etc. Again, such a process would require revisitation of our RDF-centric focused crawling techniques.

The other main challenge posed in this section is that of PLD starvation; although we would expect this to become less of an issue as the Semantic Web matures, it perhaps bears further investigation. For example, we have yet to evaluate the trade-off between small rounds with frequent updates of URIs from fresh PLDs, and large rounds which persist with a high delay-rate but require less co-ordination. Also, given the inevitability of idle time during the crawl, it may be practical to give the crawler more tasks to do in order to maximise the amount of processing done on the data, and minimise idle time.

Another aspect we have not treated in detail is that of our politeness policies: research and development of more mature politeness policies could enable a higher crawl throughput, or perhaps a more sustainable mechanism for crawling data which is in-tune with the capacity of remote data providers and competing consumers. In future, it may also be beneficial to exploit Semantic Sitemap descriptions[14] (where available) which may point to a monolithic dump of an exporter's data without the need for expensive and redundant HTTP lookups.[15]

Finally, we have not discussed the possibility of incremental crawls: choosing URIs to recrawl may lead to interesting research avenues. Besides obvious solutions such as HTTP caching, URIs could be re-crawled

[12] `http://crawler.archive.org/`; retr. 2011/01/22

[13] `http://pingthesemanticweb.com`; retr. 2011/01/22

[14] `http://sw.deri.org/2007/07/sitemapextension/`; retr. 2011/03/01

[15] However, using such Sitemaps would omit redirect information useful for later consumption of the data; also, partial crawling of a domain according to a inlink-prioritised documents would no longer be possible.

based on, e.g., detected change frequency of the document over time, some quality metric for the document, or how many times data from that document were requested in the UI. More practically, an incremental crawler could use PLD statistics derived from previous crawls, and the HTTP headers for URIs—including redirections—to achieve a much higher ratio of lookups to RDF documents returned. Such considerations would largely countermand the effects of PLD starvation, by reducing the amount of lookups the crawler needs in each run. Hand-in-hand with incremental crawling comes analysis and mechanisms for handling the dynamicity of RDF sources on the Web (e.g., see an initial survey by Umbrich et al. [2010]). For the moment, we support infrequent, independent, static crawls.

4.2 Evaluation Corpus

To obtain our evaluation Linked Data corpus, we ran the crawler continuously for 52.5 h on 8 machines from a seed list of ∼8 million URIs (extracted from an older RDF crawl) with *cur* scoring enabled.[16] In that time, we gathered a total of 1.118 billion quads, of which 11.7 million were duplicates (∼1%—representing duplicate triples being asserted in the same document).[17] We observed a mean of 140 million quads per machine and an average absolute deviation of 1.26 million across machines: considering that the average absolute deviation is ∼1% of the mean, this indicates near optimal balancing of output data on the machines.

4.2.1 Crawl Statistics

The crawl attempted 9.206 million lookups, of which 448 thousand (4.9%) were for `robots.txt` files. Of the remaining 8.758 million attempted lookups, 4.793 million (54.7%) returned response code `200 Okay`, 3.571 million (40.7%) returned a redirect response code of the form `3xx`, 235 thousand (2.7%) returned a client error code of the form `4xx` and 95 thousand (1.1%) returned a server error of the form `5xx`; 65 thousand (0.7%) were disallowed due to restrictions specified by the `robots.txt` file. Of the 4.973 million lookups returning response code `200 Okay`, 4.022 million (83.9%) returned content-type `application/rdf+xml`, 683 thousand (14.3%) returned `text/html`, 27 thousand (0.6%) returned `text/turtle`, 27 thousand (0.6%) returned `application/json`, 22 thousand (0.4%) returned `application/xml`, with the remaining 0.3% comprising of relatively small amounts of 97 other content-types—again, we only retrieve the content of the former category of documents. Of the 3.571 million redirects, 2.886 million (80.8%) were `303 See Other` as used in Linked Data to disambiguate general resources from information resources, 398 thousand (11.1%) were `301 Moved Permanently`, 285 thousand (8%) were `302 Found`, 744 (∼0%) were `307 Temporary Redirect` and 21 (∼0%) were `300 Multiple Choices`. In summary, of the non-`robots.txt` lookups, 40.7% were redirects and 45.9% were `200 Okay/application/rdf+xml` (as rewarded in our *cur* scoring mechanism). Of the 4.022 million lookups returning response code `200 Okay` and content-type `application/rdf+xml`, the content returned by 3.985 million (99.1%) were successfully parsed and included in the corpus.

An overview of the total number of URIs crawled per each hour is given in Figure 4.2; in particular, we observe a notable decrease in performance as the crawl progresses. In Figure 4.3, we give a breakdown of three categories of lookups: `200 Okay`/RDF/XML lookups, redirects, and *other*—again, our *cur* scoring views the latter category as wasted lookups. We note an initial decrease in the latter category of lookups, which then plateaus and varies between 2.2% and 8.8%.

During the crawl, we encountered 140 thousand PLDs, of which only 783 served content under `200 Okay/application/rdf+xml`. However, of the non-`robots.txt` lookups, 7.748 million (88.5%) were on the

[16] The crawl was conducted in late May, 2010.

[17] Strictly speaking, an RDF/XML document represents an RDF graph—a set of triples which cannot contain duplicates. However, given that we may sequentially access a number of very large RDF/XML documents, we parse data in streams and omit duplicate detection.

latter set of PLDs; on average, 7.21 lookups were performed on PLDs which never returned RDF/XML, whereas on average, 9,895 lookups were performed on PLDs which returned some RDF/XML. Figure 4.4 gives the number of active and new PLDs per crawl hour, where 'active PLDs' refers to those to whom a lookup was issued in that hour period, and 'new PLDs' refers to those who were newly accessed in that period; we note a high increase in PLDs at hour 20 of the crawl, where a large amount of 'non-RDF/XML PLDs' were discovered. Perhaps giving a better indication of the nature of PLD starvation, Figure 4.5 renders the same information for only those PLDs who return some RDF/XML, showing that half of said PLDs are exhausted after the third hour of the crawl, that only a small number of new 'RDF/XML PLDs' are discovered after the third hour (between 0 and 14 each hour), and that the set of active PLDs plateaus at ~50 towards the end of the crawl.

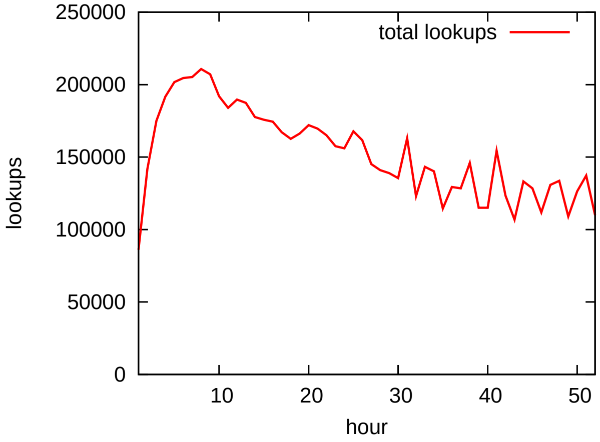


Figure 4.2: Number of HTTP lookups per crawl hour.

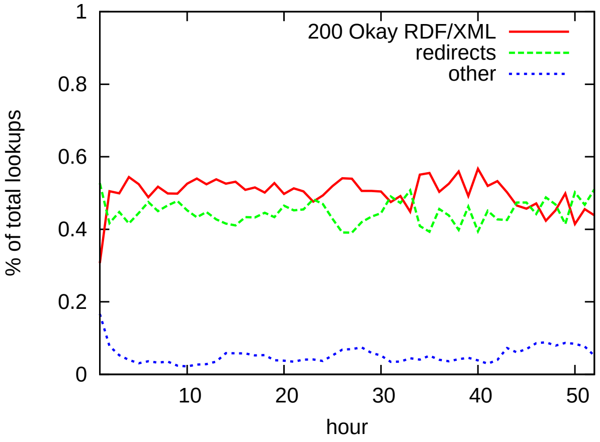


Figure 4.3: Breakdown of HTTP lookups per crawl hour.

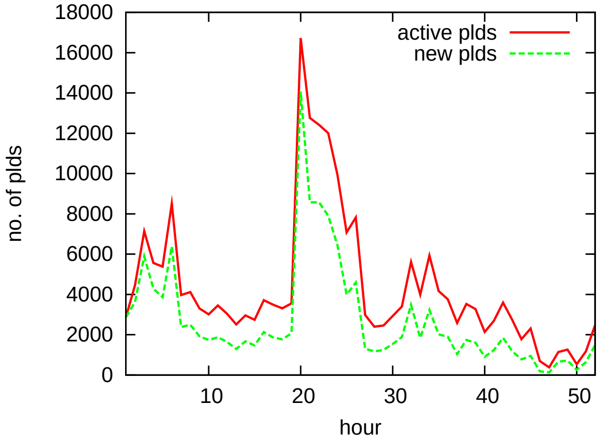


Figure 4.4: Breakdown of PLDs per crawl hour.

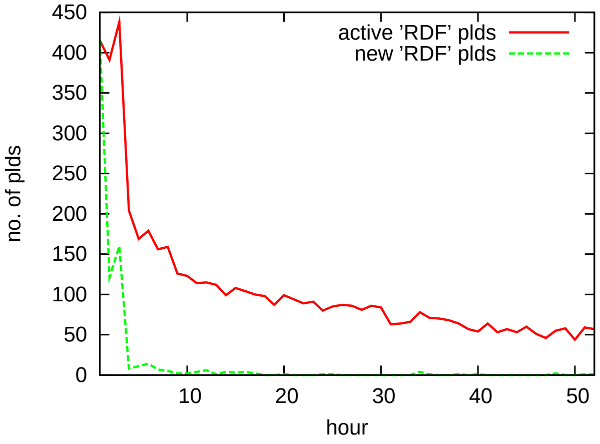


Figure 4.5: Breakdown of RDF/XML PLDs per crawl hour.

4.2.2 Corpus Statistics

The resulting evaluation corpus is sourced from 3.985 million documents and contains 1.118 billion quads, of which 1.106 billion are unique (98.9%) and 947 million are unique triples (84.7% of raw quads).

To characterise our corpus, we first look at a breakdown of data providers. We extracted the PLDs from the source documents and summated occurrences: Table 4.3 shows the top 25 PLDs with respect to the number of triples they provide in our corpus, as well as their document count and average number of triples per document. We see that a large portion of the data is sourced from social networking sites—such as hi5.com and livejournal.com—that host FOAF exports for millions of users. Notably, the hi5.com domain provides 595 million (53.2%) of all quadruples in the data: although the number of documents crawled from this domain was comparable with other high yield domains, the high ratio of triples per document meant that in terms of quadruples, hi5.com provides the majority of data. Other sources in the top-5 include the opiumfield.com domain which offers LastFM exports, and linkedlifedata.com and bio2rdf.org which publishes data from the life-science domain.

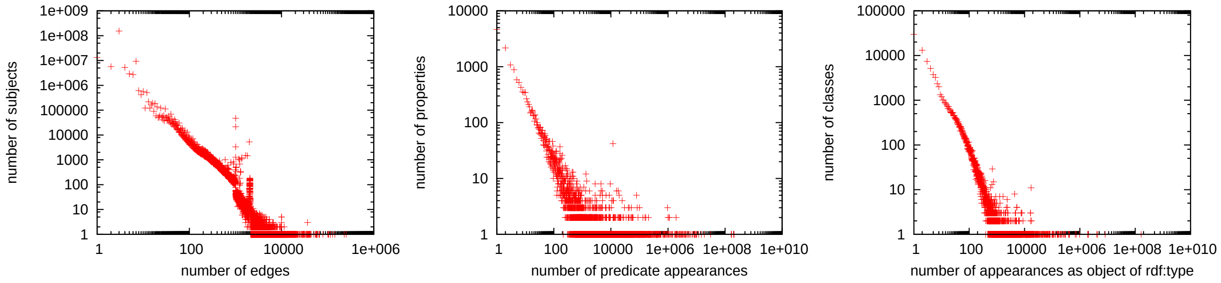
#	PLD	quads	documents	quads/document
1	hi5.com	595,086,038	255,742	2,327
2	livejournal.com	77,711,072	56,043	1,387
3	opiumfield.com	66,092,693	272,189	243
4	linkedlifedata.com	54,903,837	253,392	217
5	bio2rdf.org	50,659,976	227,307	223
6	rdfize.com	38,107,882	161,931	235
7	appspot.com	28,734,556	49,871	576
8	identi.ca	22,873,875	65,235	351
9	freebase.com	18,567,723	181,612	102
10	rdfabout.com	16,539,018	135,288	122
11	ontologycentral.com	15,981,580	1,080	14,798
12	opera.com	14,045,764	82,843	170
13	dbpedia.org	13,126,425	144,850	91
14	qdos.com	11,234,120	14,360	782
15	l3s.de	8,341,294	163,240	51
16	dbtropes.org	7,366,775	34,013	217
17	uniprot.org	7,298,798	11,677	625
18	dbtune.org	6,208,238	181,016	34
19	vox.com	5,327,924	44,388	120
20	bbc.co.uk	4,287,872	262,021	16
21	geonames.org	4,001,272	213,061	19
22	ontologyportal.org	3,483,480	2	1,741,740
23	ordnancesurvey.co.uk	2,880,492	43,676	66
24	loc.gov	2,537,456	166,652	15
25	fu-berlin.de	2,454,839	135,539	18

Table 4.3: Top twenty-five PLDs and number of quads they provide.

Continuing, we encountered 199.4 million unique subjects, of which 165.3 million (82.9%) are blank-nodes and 34.1 million (17.1%) are URIs. This translates into an average of 5.6 quadruple occurrences per subject; Figure 4.6(a) shows the distribution of the number of quadruple appearances for each unique subject, where one subject appears in 252 thousand quadruples, and where the plurality of subjects occur in three quadruples—a power-law is *ostensibly* apparent.[18] There are some irregular "peaks" apparent the distribution, where we notice large groups of subjects have precisely the same number of edges due to triple

[18] We are reluctant to *claim* a power-law following the criticism of Clauset et al. [2009] with respect to the over-prevalence of such claims in research literature.

limits enforced by certain exporters (we will discuss this further in § D.4 where it affects our evaluation).



(a) Subjects (b) Predicates (c) `rdf:type` values

Figure 4.6: Distributions for number of triples per (a) subject, (b) predicate (properties), (c) value of `rdf:type` (classes) [log/log].

With respect to objects, we found that 668 million are URIs (60.4%), 273.5 million are literals (24.7%), and 164.5 million (14.9%) are blank-nodes.

Next, we look at usage of properties and classes in the data: for properties, we analysed the frequency of occurrence of terms in the predicate position, and for classes, we analysed the occurrences of terms in the object position of `rdf:type` quads. We found 23,155 unique predicates, translating into an average 48,367 quads per predicate; Figure 4.6(b) gives the distribution of predicate occurrences (property usage), which again resembles a power-law, and where the plurality of predicates (4,659 | 20.1%) appear only once; Table 4.4 gives the listing of the top twenty-five predicates, where (unsurprisingly) `rdf:type` heads the list (18.5% of all quads), and where `foaf` properties also feature prominently.

Analogously, we found 104,596 unique values for `rdf:type`, translating into an average of 1,977 `rdf:type` quadruples per class term; Figure 4.6(c) gives the distribution of `rdf:type`-value occurrences (class usage), which also *resembles* a power-law, where the plurality of classes (29,856 | 28.5%) appear only once; Table 4.4 gives the listing of the top twenty-five classes, where again FOAF—and in particular `foaf:Person` (79.2% of all `rdf:type` quads)—features prominently.

In order to get an insight into the most instantiated vocabularies, we extracted the 'namespace' from predicates and URI-values for `rdf:type`—we simply strip the URI upto the last hash or slash. Table 4.4 gives the top-25 occurring namespaces for a cumulative count, where FOAF, RDFS, and RDF dominate; in contrast, Table 4.4 also gives the top 25 namespaces for unique URIs appearing as predicate or value of `rdf:type`, where in particular namespaces relating to DBPedia, Yago and Freebase offer a diverse set of *instantiated* terms; note that (i) the terms need not be defined in that namespace (e.g., `foaf:tagLine` used by LiveJournal) or may be misspelt versions of defined terms (e.g., `foaf:image` used by LiveJournal instead of `foaf:img` [Hogan et al., 2010a]), and (ii) 460 of the 489 terms in the `rdf:` namespace are predicates of the form `rdf:`_*n*.

4.2.3 Related Work

With respect to evaluation corpora, Semantic Web research has long relied upon synthetic benchmarks to demonstrate (in particular) the performance of various techniques for reasoning, indexing, query answering, etc.

Perhaps the most prominent synthetic frawework for evaluating Semantic Web systems is the "Lehigh University Benchmark" (LUBM) proposed by Guo et al. [2005]—in fact, we will use this benchmark ourselves for some initial performance evaluation of reasoning optimisations in § 5.3.4. The LUBM framework consists

#	predicate	triples	class	triples	namespace	triples	namespace	terms
1	rdf:type	206,799,100	foaf:Person	163,699,161	foaf:	615,110,022	yagor:	41,483
2	rdfs:seeAlso	199,957,728	foaf:Agent	8,165,989	rdfs:	219,205,911	yago:	33,499
3	foaf:knows	168,512,114	skos:Concept	4,402,201	rdf:	213,652,227	dbtropes:	16,401
4	foaf:nick	163,318,560	mo:MusicArtist	4,050,837	b2r:/b2rr:	43,182,736	fb:	14,884
5	b2rr:linkedToFrom	31,100,922	foaf:PersonalProfileDocument	2,029,533	lldpubmed:	27,944,794	dbp:	7,176
6	lldegene:pubmed	18,776,328	foaf:OnlineAccount	1,985,390	lldegene:	22,228,436	own16:	1,149
7	rdfs:label	14,736,014	foaf:Image	1,951,773	skos:	19,870,999	semwebid:	1,024
8	owl:sameAs	11,928,308	opiumfield:Neighbour	1,920,992	fb:	17,500,405	opencyc:	937
9	foaf:name	10,192,187	geonames:Feature	983,800	owl:	13,140,895	estoc:	927
10	foaf:weblog	10,061,003	foaf:Document	745,393	opiumfield:	11,594,699	dbo:	843
11	foaf:homepage	9,522,912	owl:Thing	679,520	mo:	11,322,417	sumo:	581
12	lldpubmed:chemical	8,910,937	estoc:cphi_m	421,193	dc:	9,238,140	rdf:	489
13	foaf:member_name	8,780,863	gr:ProductOrServiceModel	407,327	estoc:	9,175,574	wn:	303
14	foaf:tagLine	8,780,817	mo:Performance	392,416	dct:	6,400,202	b2r:/b2rr:	366
15	foaf:depiction	8,475,063	fb:film.performance	300,608	b2rns:	5,839,771	movie:	251
16	foaf:image	8,383,510	fb:tv.tv_guest_role	290,246	sioc:	5,411,725	uniprot:	207
17	foaf:maker	7,457,837	fb:common.webpage	288,537	vote:	4,057,450	aifb:	196
18	lldpubmed:journal	6,917,754	dcmit:Text	262,517	geonames:	3,985,276	factbook:	191
19	foaf:topic	6,163,769	estoc:irt_h_euryld_d	243,074	skipinions:	3,466,560	foaf:	150
20	lldpubmed:keyword	5,560,144	owl:Class	217,334	dbo:	3,299,442	geospecies:	143
21	dc:title	5,346,271	opwn:WordSense	206,941	uniprot:	2,964,084	b2rns:	133
22	foaf:page	4,923,026	bill:LegislativeAction	193,202	estoc:	2,630,198	ludopinions:	132
23	b2r:linkedToFrom	4,510,169	fb:common.topic	190,434	lldlifeskim:	2,603,123	oplweb:	130
24	skos:subject	4,158,905	rdf:Statement	169,376	ptime:	2,519,543	esns:	127
25	skos:prefLabel	4,140,048	mu:Venue	166,374	dbp:	2,371,396	drugbank:	119

Table 4.4: Top twenty-five (i) predicates by number of triples they appear in; (ii) values for `rdf:type` by number of triples they appear in; and (iii) namespaces by number of triples the contained predicates or values for `rdf:type` appear in; (iv) namespaces by unique predicates or values for `rdf:type` that they contain.

of (i) an OWL ontology which provides a formally defined terminology relating to universities, staff, students, and so forth, (ii) code for populating variable-sized corpora of instance data using this terminology and pseudo-random generative algorithms, (iii) a set of benchmark (SPARQL [Prud'hommeaux and Seaborne, 2008]) queries of various types which can be posed against the corpora, some of which rely on inferencing to generate a complete set of answers. LUBM has been heavily used for scalability benchmarks given that massive corpora can be quickly generated for testing; e.g., see the recent evaluation of reasoning systems by Weaver and Hendler [2009]; Urbani et al. [2009, 2010] and evaluation of indexing and query-processing by ourselves [Harth et al., 2007], as well as Liu and Hu [2005]; Erling and Mikhailov [2009]; Stocker et al. [2008], etc.

Ma et al. [2006] later extended LUBM to create the "University Ontology Benchmark" (UOBM), introducing variants of the university ontology which make more extensive use of the OWL Lite and OWL DL dialects. They also noted that the LUBM benchmark was poorly interlinked on a global level, with data from different universities forming (in our words) "cliques"; they thus updated the generative algorithms to better interlink different universities. UOBM was subsequently used to evaluate various reasoning engines, such as those proposed by Zhou et al. [2006]; Lu et al. [2007].

More recently, Bizer and Schultz [2009] proposed the "Berlin SPARQL Benchmark" (BSBM): a framework for evaluating SPARQL query engines based on synthetic e-commerce data and a set of benchmark queries—unlike LUBM, reasoning is not part of the benchmark. BSBM has been used in a number of evaluations, including for systems proposed by Castillo and Leser [2010]; Martin et al. [2010] etc.

A number of other benchmarking suites have also been proposed, but—to the best of our knowledge—have yet to enjoy notable adoption, including the "Lehigh Bibtex Benchmark" (LBBM) [Wang et al., 2005] and SP^{2}Bench for SPARQL query-answering [Schmidt et al., 2009a].

However, synthetic benchmarks are inherently unsuitable for evaluating systems which intend to operate over real-world and/or heterogeneous corpora, given that (i) the morphology of the data created by the generative algorithms may be overly simplistic and not reflective of real data; (ii) the queries defined may not align with those of a typical use-case; (iii) the data typically correspond to a closed, single terminology; (iv) the data can always be assumed to be correct. Acknowledging these concerns, numerous other real-world, curated datasets have been used in evaluating various Semantic Web systems, including UniProt [Wu et al., 2006], DBPedia [Bizer et al., 2009b], OpenCyc[19], etc. Again, although these datasets may feature some noise, and may be the result of collaborative editing and curation, they are not representative of heterogeneous Web corpora.

Moving towards evaluation of more heterogeneous data, Kiryakov et al. [2009] collected together a selection of dumps hosted by various Linked Data providers—viz., DBPedia, Freebase[20], Geonames[21], UMBEL[22], WordNet®[23], CIA World Factbook[24], Lingvoj[25], MusicBrainz[26]—and selected vocabularies—viz. Dublin Core, SKOS, RSS and FOAF—to create a corpus of one billion facts for the "Linked Data Semantic Repository" (LDSR). Unlike our corpus, the sources of data constituting LDSR are manually chosen; although the authors discuss methods to overcome various forms of domain-specific noise present in their data, they do not address generic issues of provenance or handling arbitrary Web data.

Again, like us, various systems crawl their own corpora for populating and evaluating their systems; please see discussion in § 4.1.7.

[19] `http://www.cyc.com/cyc/opencyc`; retr. 2010/01/10

[20] `http://www.freebase.com/`; retr. 2010/01/10

[21] `http://www.geonames.org/`; retr. 2010/01/10

[22] `http://www.umbel.org/`; retr. 2010/01/10

[23] `http://wordnet.princeton.edu/`; retr. 2010/01/10

[24] `http://www4.wiwiss.fu-berlin.de/factbook/`; retr. 2010/01/10

[25] `http://www.lingvoj.org/`; retr. 2010/01/10

[26] `http://musicbrainz.org/`; retr. 2010/01/10

With respect to evaluating systems against real-world RDF Web Data, perhaps the most agreed upon corpus is that provided annually for the"Billion Triple Challenge" (BTC)[27]: this corpus is crawled every year from millions of sources, and entrants to the challenge must demonstrate applications thereover. Since the first challenge in 2008, a number of other papers have used BTC corpora (or some derivation there from) for evaluation, including works by Erling and Mikhailov [2009]; Urbani et al. [2009]; Schenk et al. [2009]; Delbru et al. [2010a], etc.—-as well as ourselves [Hogan et al., 2009b] and (of course) various other entrants to the BTC itself.[28]

4.2.4 Critical Discussion and Future Directions

Firstly, we again note that our corpus only consists of RDF/XML syntax data, and thus we miss potentially interesting contributions from, in particular, publishers of RDFa—for example, GoodRelations data [Hepp, 2009] is often published in the latter format. However, we conjecture that RDF/XML is still the most popular format for Linked Data publishing, and that only considering RDF/XML still offers a high coverage of those RDF providers on the Web.[29]

Further, we note that some of the sources contributing data to our corpus may not be considered Linked Data in the strict sense of the term: some RDF exporters—such as `opera.com`—predate the Linked Data principles, and may demonstrate (i) sparse use of URIs (**LDP1**/**LDP2**/**LDP3**), and (ii) sparse outlinks to external data sources (**LDP4**). However, these exporters are published as RDF/XML on the Web, receive inlinks from other Linked Data sources, and often share a common vocabulary—particularly FOAF—with other Linked Data providers; since we do not want to blacklist providers of RDF/XML Web documents, we consider the data provided by these exporters as "honourary" Linked Data.

Similarly, a large fragment of our corpus is sourced from FOAF exporters which provide uniform data and which we believe to be of little general interest to users outside of that particular site—again, such dominance in data volume is due to a relatively large triple-to-document ratio exhibited by domains such as `hi5.com` (cf. Table 4.3). In future, we may consider a triple-based budgeting of domains to ensure a more even triple count across all data providers, or possibly PLD prioritisation according to inlink-based quality measures (it's worth noting, however, that such measures would make our crawl sub-optimal with respect to our politeness policies).

With respect to the scale of our corpus—in the order of a billion triples—we are still (at least) an order of magnitude below the current amount of RDF available on the Web of Data. Although our methods are designed to scale beyond the billion triple mark, we see the current scope as being more exploratory, and evaluating the wide variety of algorithms and techniques we present in later chapters for a larger corpus would pose significant practical problems, particularly with respect to our hardware and required runtimes. In any case, we believe that our billion triple corpus poses a very substantial challenge with respect to the efficiency and scalability of our methods—later chapters include frank discussion on potential issues with respect to scaling further (these issues are also summarised in Chapter 8).

Finally, given that we perform crawling of real-world data from the Web, we do not have any form of gold-standard against which to evaluate our methods—this poses inherent difficulties with respect to evaluating the quality of results. Thus, we rely on known techniques rooted in the standards by which the data are published, such that errors in our results are traceable to errors in the data (with respect to the standards) and not deficiencies in our approach. Additionally, we offer methods for automatically detecting noise in our

[27] `http://challenge.semanticweb.org/`; retr. 2010/01/10

[28] In fact, these corpora have been crawled by colleagues Andreas Harth and Jürgen Umbrich using SWSE architecture.

[29] Note that at the time of writing, Drupal 7 has just been released, offering RDFa export of data as standard; for example, see `http://semanticweb.com/drupal-7-debuts-parties-set-to-begin_b17277`; retr. 2010/01/10. We hope that these developments will lead to richer sources of RDF in future.

results—in the form of inconsistency—and sketch approaches for repairing such problems.

4.3 Ranking

We now give an overview of our distributed implementation for ranking Linked Data sources, based on the methods of Harth et al. [2009]; again, we will use the results of this ranking procedure in Chapter 5 when analysing the use of RDFS and OWL in Linked Data, and also in Chapter 6 as the basis for annotating triples with ranking scores.

4.3.1 Rationale and High-level Approach

There is a long history of links-based analysis over Web data—and in particular over hypertext documents—where links are seen as a positive vote for the relevance or importance of a given document. Seminal works exploiting the link structure of the Web for ranking documents include HITS [Kleinberg, 1999] and PageRank [Page et al., 1998]; however, these techniques are not immediately applicable to Linked Data. More recent works (e.g., [Harth et al., 2009; Delbru et al., 2010b]) have presented various links-based techniques for ranking RDF corpora with various end-goals in mind: most commonly, prioritisation of informational artefacts in user result-views.

Inspired in particular by work presented in [Harth et al., 2009], herein we use links-based analysis with the underlying premise that higher ranked sources contribute more "trustworthy" data: in our case, we *assume* a correlation between the (Eigenvector) centrality of a source in the graph, and the "quality" of data that it provides.[30] We thus implement a two-step process similar to that presented in [Harth et al., 2009]: (i) we create the graph of links between sources, and apply a standard PageRank calculation over said graph to derive source ranks; (ii) we propagate source ranks to the *elements of interest* they contain using a simple summation aggregation—elements of interest may be terms, triples, predicates, axioms, etc., as required. Herein, we focus on the first step: viz., ranking sources.

Harth et al. [2009] discuss and compare two levels of granularity for deriving source-level ranks: (i) document-level granularity analysing inter-linkage between Web documents; and (ii) *pay-level-domain* (PLD) granularity analysing inter-linkage between domains (e.g., `dbpedia.org`, `data.gov.uk`). Document-level analysis is more expensive—in particular, it generates a larger graph for analysis—but allows for more fine-grained ranking of elements resident in different documents. PLD-level analysis is cheaper—it generates a smaller graph—but ranks are more coarse-grained, and many elements can share identical ranks grouped by the PLDs they appear in.

In the SWSE engine [Hogan et al., 2010b], we employ in-memory techniques for applying a PageRank analysis of the PLD-granularity source graph: we demonstrated this to be efficiently computable in our distributed setup due to the small size of the graph extracted (in particular, the PageRank calculation took <1 min in memory applying 10 iterations), and argued that PLD-level granularity was sufficient for that particular use-case (ranking entity 'importance', which is combined with TF-IDF relevance scores for prioritising entity-search results) [Hogan et al., 2010b]. However, in this thesis, we opt to apply the more expensive document-level analysis which provides more granular results useful, in particular, for repairing inconsistent data in § 6.4.3. Since we will deal with larger graphs, we opt for on-disk batch-processing techniques—mainly sorts, scans, and sort-merge joins of on-disk files—which are not hard-bound by in-memory capacity, but which are significantly slower than in-memory analysis.

[30] With respect to Linked Data, data quality is a complex issue worthy of a thesis itself—we leave the definition open and the word "quality" in inverted commas.

4.3.2 Creating and Ranking the Source Graph

Creating the graph of interlinking Linked Data sources is non-trivial, in that the notion of a hyperlink does not directly exist for our corpus. Thus, we must extract a graph sympathetic to Linked Data principles (§ 3.3) and current publishing patterns.

According to **LDP4**, links should be specified simply by using external URI names in the data. These URI names should dereference to an RDF description of themselves according to **LDP2** and **LDP3** respectively. Let $D = (V, E)$ represent a simple directed graph where $V \subset \mathsf{S}$ is a set of sources (vertices), and $E \subset \mathsf{S} \times \mathsf{S}$ is a set of pairs of vertices (edges). Letting $s_i, s_j \in V$ be two vertices, then $(s_i, s_j) \in E$ iff $s_i \neq s_j$ and there exists some $u \in \mathsf{U}$ such that $\mathsf{redirs}(u) = s_j$ and u appears in some triple $t \in \mathsf{get}(s_i)$: i.e., an edge extends from s_i to s_j iff the RDF graph returned by s_i mentions a URI which dereferences to a different source s_j.

Now, let $E(s)$ denote the set of direct successors of s (outlinks), let $E_\emptyset$ denote the set of vertices with no outlinks (dangling nodes), and let $E^-(s)$ denote the set of direct predecessors of s (inlinks). The PageRank of a vertex s_i in the directed graph $D = (V, E)$ is then given as follows [Page et al., 1998]:

$$\mathsf{rank}(s_i) = \frac{1-d}{|V|} + d \sum_{s_\emptyset \in E_\emptyset} \frac{\mathsf{rank}(s_\emptyset)}{|V|} + d \sum_{s_j \in E^-(s_i)} \frac{\mathsf{rank}(s_j)}{|E(s_j)|}$$

where d is a damping constant (usually set to 0.85) which helps ensure convergence in the following iterative calculation, and where the middle component splits the ranks of dangling nodes evenly across all other nodes. Note also that the first and second components are independent of i, and constitute the minimum possible rank of all nodes (ensures that ranks do not need to be normalised during iterative calculation).

Now let $w = \frac{1-d}{|V|}$ represent the weight of a universal (weak link) given by all non-dangling nodes to all other nodes—dangling nodes split their vote evenly and thus don't require a weak link; we can use a weighted adjacency matrix M as follows to encode the graph $D = (V, E)$:

$$m_{i,j} = \begin{cases} \frac{d}{|E(s_j)|} + w, & \text{if } (s_j, s_i) \in E \\ \frac{1}{|V|}, & \text{if } s_j \in E_\emptyset \\ w, & \text{otherwise} \end{cases}$$

where this stochastic matrix can be thought of as a Markov chain (dubbed the random-surfer model). The ranks of all sources can be expressed algebraically as the principal eigenvector of M, which in turn can be estimated using the power iteration method up until some termination criterium (fixed number of iterations, convergence measures, etc.) is reached. We refer the interested reader to [Page et al., 1998] for more detail.

4.3.3 Distributed Ranking Implementation

Individual tasks are computed using parallel sorts, scans and merge-joins of flat compressed files over the slave machines in our cluster; we assume that the corpus is pre-distributed over the cluster, as is the output of our crawling component. For reference, we provide the detailed description of the ranking sub-algorithms in Appendix C, where our high-level distributed approach can be *summarised* as follows:

1. **run**: each slave machine sorts its segment of the data by context (aka. *source*), with a list of sorted contexts generated on each machine;

2. **gather/flood**: the master machine gathers the list of contexts, merge-sorting the raw data into a global list of contexts which is subsequently flooded to the slave machines;

3. **run**: the slave machines extract the source-level graph in parallel from their segment of the data (Algorithm C.1), rewrite the vertices in the sub-graph using redirects (Algorithm C.2), prune links in

#	Document	Rank
1	http://www.w3.org/1999/02/22-rdf-syntax-ns	0.112
2	http://www.w3.org/2000/01/rdf-schema	0.104
3	http://dublincore.org/2008/01/14/dcelements.rdf	0.089
4	http://www.w3.org/2002/07/owl	0.067
5	http://www.w3.org/2000/01/rdf-schema-more	0.045
6	http://dublincore.org/2008/01/14/dcterms.rdf	0.032
7	http://www.w3.org/2009/08/skos-reference/skos.rdf	0.028
8	http://www.w3.org/2003/g/data-view	0.014
9	http://xmlns.com/foaf/spec/	0.014
10	http://www.w3.org/2000/01/combined-ns-translation.rdf.fr	0.010

Table 4.5: Top 10 ranked documents

the sub-graph such that it only contains vertices in the global list of contexts (Algorithm C.3), and finally sort the sub-graph according to inlinks and outlinks;

4. **gather/flood**: the master machine gathers the sub-graphs (for both inlink and outlink order) from the slave machines and merge-sorts to create the global source-level graph; the master machine then performs the power iteration algorithm to derive PageRank scores for individual contexts using on-disk sorts and scans (Algorithms C.4 & C.5); ranks are subsequently flooded to each machine;

The result of this process is context rank pairs of the form (c, r) available on disk to all machines.

4.3.4 Ranking Evaluation and Results

Applying distributed ranking over the 1.118 billion statement corpus (which is pre-distributed over the slave machines as a result of the crawl) took just under 30.3 h, with the bulk of time consumed as follows: (i) parallel sorting of data by context took 2.2 h; (ii) parallel extraction and preparation of the source-level graph took 1.9 hours; (iii) ranking the source-level graph on the master machine took 26.1 h (applying ten power iterations). In particular, the local on-disk PageRank calculation—performed on the master machine—proved the most expensive task, and would become a bottleneck when increasing slave machines.[31]

The source-level graph consisted of 3.985 million vertices (sources) and 183 million unique non-reflexive links. In Table 4.5, we provide the top 10 ranked documents in our corpus. The top result refers to the RDF namespace, followed by the RDFS namespace, the DC namespace and the OWL namespace (the latter three are referenced by the RDF namespace, and amongst themselves). Subsequently, the fifth result contains some links to multi-lingual translations of labels for RDFS terms, and is linked to by the RDFS document; the sixth result refers to an older version of DC (extended by result 3); the seventh result is the SKOS vocabulary; the eighth result provides data and terms relating to the GRDDL W3C standard; the ninth result refers to the FOAF vocabulary; the tenth is a French translation of RDFS terms. Within the top ten, all of the documents are somehow concerned with terminological data (e.g., are ontologies or vocabularies, or—sometimes informally—describe classes or properties): the most wide-spread reuse of terms across the Web of Data is on a terminological level, representing a higher in-degree and subsequent rank for terminological documents (cf. § 4.2.2).

4.3.5 Related Work

Applying links-based analysis to Web data has its roots in the seminal approaches of HITS [Kleinberg, 1999] and PageRank [Page et al., 1998] which have spawned a rich area of research. Herein, we briefly introduce

[31] In future, it would be worthwhile investigate distribution of this task; for example, see [Gleich et al., 2004].

proposals which—like us—specifically aim at ranking RDF.

Balmin et al. [2004] introduced "ObjectRank" for applying PageRank over a directed labelled graph using "authority transfer schema graphs", which requires manual weightings for the transfer of propagation through different types of links; further, the algorithm does not include consideration of the source of data, and is perhaps better suited to domain-specific ranking over verified knowledge.

The Swoogle RDF search engine ranks documents using the "OntoRank" method [Ding et al., 2005], a variation on PageRank which iteratively calculates ranks for documents based on references to terms (classes and properties) defined in other documents. However, they rely on manually specified weightings for links between documents, and since they predate Linked Data, do not consider dereferenceability of terms in constructing the graph.

We previously introduced "ReConRank" [Hogan et al., 2006]: an initial effort to apply a PageRank-type algorithm to a graph which unifies data-level and source-level linkage. ReConRank does take data provenance into account: however, because it simultaneously operates on the data-level graph, it is more susceptible to spamming than the presented approach.

A more recent proposal by [Delbru et al., 2010b] for ranking Linked Data—called "Dataset rankING" (DING)—holds a similar philosophy to that presented herein: they adopt a two-layer approach consisting of an entity layer and a dataset layer. However, they also apply rankings of entities within a given dataset, using PageRank (or optionally link-counting) and unsupervised link-weighting schemes, subsequently combining dataset and local entity ranks to derive global entity ranks.

There are numerous other *loosely* related RDF ranking approaches which—although they do not deal directly with ranking sources—we briefly mention. The Ontocopi system [Alani et al., 2003] uses a spreading activation algorithm to locate instances in a knowledge base which are most closely related to a target instance; later, Alani et al. [2006] introduced "AKTiveRank" for ranking ontologies according to how well they cover specified search terms. "SemRank" [Anyanwu et al., 2005] ranks relations and paths on Semantic Web data using information-theoretic measures. The SemSearch system [Lei et al., 2006] also includes relevance ranks for entities according to how well they match a user query.

4.3.6 Critical Discussion and Future Directions

Although links-based analyses have proven themselves fruitful for ranking HTML documents, we can only conjecture a correlation between the Eigenvector centrality of Linked Data documents and the quality or reliability of the data they provide. We note that Harth et al. [2009] performed a user-evaluation to validate the outlined ranking procedure for prioritising results as they appear in the SWSE system, but herein we use the ranking scores as an (unproven) indicator of reliability of data. Still—and as per the HTML Web—we believe that well-linked RDF documents should intuitively provide higher-quality information. Although more intricate methods for constructing or ranking the source-level graph may provide better results for our purposes, we leave investigation thereof for a necessarily more focused scope.

Similarly, Google's use of the PageRank algorithm encouraged the proliferation of HTML "link farms": heavily interlinked groups of artifice websites used to game the PageRank calculations and promote certain results for popular search terms. This precedent demonstrates that the naïve PageRank model is not tolerant to certain forms of spamming. Currently, there is insufficient economic or social motivation for such spamming on the Web of Data, and although (as we will see) substantial noise is present, we believe that it is primarily indeliberate. As the Web of Data matures, so too must various algorithms operating over it.

Finally, using only one machine to apply the PageRank calculations over the source-level graph poses a practical barrier to efficiency and scale; in future we would reasonably prioritise pushing the execution of this task onto the slave machines—indeed, parallelising the underlying sorts and scans may prove sufficient.

Chapter 5

Reasoning[*]

> *"The question of whether computers can think is just like the question of whether submarines can swim."*
>
> —**Edsger W. Dijkstra**

Given a sufficiently large corpus from the Web, heterogeneity in terminology often poses a significant problem to its subsequent consumption. 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` defines the relationship from resources to the documents somehow concerning them, we can formulate a simple structured query in SPARQL [Prud'hommeaux and Seaborne, 2008]—the W3C standardised RDF query language—as given in Listing 5.1.

Listing 5.1: Simple query for all pages relating to `ex:resource`

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

However, in our corpus there exist 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 5.2 involving properties in our corpus, where 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

[*] Parts of this chapter have been published as [Hogan et al., 2008, 2009b; Hogan and Decker, 2009; Hogan et al., 2010c].

Listing 5.2: 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
  ...
}
```

properties referenced in Listing 5.2 are either 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 5.2. Taking some examples, `plink:profile` is defined to have the relation `rdfs:subPropertyOf` to `foaf:page`; `doap:homepage` is defined to have the relation `rdfs:subPropertyOf` to `foaf:homepage`, which itself has the relation `rdfs:subPropertyOf` to `foaf:isPrimaryTopicOf`, which in turn has the `rdfs:subPropertyOf` relation to `foaf:page`; `foaf:topic` has the relation `owl:inverseOf` to `foaf:page`; and so forth.[1] Thus, 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 *reasoning.*

Herein, we detail our system for performing reasoning over large-scale Linked Data corpora, which we call the *Scalable Authoritative OWL Reasoner* (SAOR); in particular:

- we begin by discussing the requirements of our system for performing Linked Data/Web reasoning and high-level design decisions (§ 5.1);
- we continue by discussing a separation of terminological data during reasoning, providing soundness and conditional completeness results (§ 5.2);
- we subsequently detail the generic optimisations that a separation of terminological data allows (§ 5.3);
- 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 (§ 5.4);
- finally, we provide an overview of related work (§ 5.5) and give general discussion (§ 5.6).

5.1 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.

5.1.1 Incomplete Reasoning: Rationale

Following discussion in § 2.2, standard 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 [ter Horst, 2005b; Muñoz et al., 2007, 2009; Weaver and Hendler, 2009]. In any case, RDFS does not support reasoning over OWL axioms commonly provided by Linked Data vocabularies.

[1] Of course, other relevant properties may not be related to `foaf:page`, but we would expect certain self-organising phenomena in the community to ensure that commonly instantiated terms (which provide the most answers) are properly interlinked with other commonly instantiated terms, as has been encouraged within the Linked Data community.

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 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 [Baader et al., 2002]. 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). Some older surveys of Web data echo this claim: Bechhofer and Volz [2004] identified and categorised OWL DL restrictions violated by a sample group of 201 OWL ontologies (all of which were found to be in OWL Full); these include incorrect or missing typing of classes and properties, complex object-properties (e.g., functional properties) declared to be transitive, inverse-functional datatype properties, etc. Wang et al. [2006a] later provided a more extensive survey over ∼1,300 ontologies: 924 were identified as being in OWL Full. However, it is worth noting that in both surveys, the majority of infringements into OWL Full were found to be purely syntactic, rather innocuous with no real effect on decidability, and unambiguously repairable without affecting expressivity; after repair, Wang et al. [2006b] showed that the majority of Web documents surveyed were within the relatively inexpressive $\mathcal{ALC}$ Description Logic.

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 use the results of Horrocks and Patel-Schneider [2004] to 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 (that is, the entire set G) from our knowledge-base. Along these lines, a number of paraconsistent reasoning approaches have been defined in the literature (see, e.g., [Huang and van Harmelen, 2008; Ma and Hitzler, 2009; Zhang et al., 2009; Maier, 2010; Lembo et al., 2010]) typically relying upon four-valued logic [Belnap, 1977][2]—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. Thereafter, one could consider distilling a sub-language by which complete reasoning becomes feasible for the Web (it would seem that this language could not allow inconsistency)—although we admit that this is an interesting formal exercise, we consider it as outside of the current scope, and argue that in practice, it would serve to restrict the expressiveness available to vocabulary publishers when defining their terms, drawing analogy to FOAF's (reasonable) definition of `foaf:mbox_sha1sum` as being simultaneously a

[2] The four values are: `true`, `false`, `neither/undefined/unknown`, `both/overdefined/inconsistent`.

datatype-property and an inverse-functional-property breaching current OWL DL syntactic restrictions.[3]

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; this is argued by Fensel and van Harmelen [2007] who state:

> *"The Web is open, with no defined boundaries. Therefore, completeness is a rather strange requirement for an inference procedure in this context. Given that collecting all relevant information on the Web is neither possible nor often even desirable (usually, you want to read the first 10 Google hits but dont have the time for the remaining two million), requesting complete reasoning on top of such already heavily incomplete facts seems meaningless."*
>
> **—Fensel and van Harmelen [2007]**

Similar arguments for incomplete reasoning in scenarios such as ours are laid out by Hitzler and van Harmelen [2010] who state:

> "[...] *we would advocate* [viewing] *the formal semantics of a system (in whatever form it is specified) as a "gold standard", that need not necessarily be obtained in a system (or even be obtainable). What is required from systems is not a proof that they satisfy this gold standard, but rather a precise description of the extent to which they satisfy this gold standard."*
>
> **—Hitzler and van Harmelen [2010]**

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).

5.1.2 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.

One of the earliest efforts looking at combining rules and ontologies was Description Logic Programs (DLP) introduced by Grosof et al. [2004] and refined by Motik [2004]. The key aim of DLP is to identify and characterise the expressive intersection of the Description Logics (DL) and Logic Program (LP) formalisms, such that knowledge represented in DLP can be mapped to either formalism without loss of meaning, and thus is amenable to both tableau and rule-based implementation of inferencing. Since DLP predates OWL, the authors discussed how individual RDFS and DAML+OIL constructs are expressible in DL and LP and under what syntactic restrictions. These results carry naturally over to OWL, and lead to a restricted, rule-expressible fragment of OWL DL, which includes full support for `owl:inverseOf`, `owl:TransitiveProperty`, `owl:equivalentProperty`, `owl:equivalentClass`, `owl:intersectionOf`, and restricted support for `owl:unionOf`, `owl:someValuesFrom`, and `owl:allValuesFrom`. Following this work, de Bruijn et al. [2005b] subsequently proposed the OWL$^-$ family of OWL dialects, defining rule-expressible syntactic subsets of OWL Lite, OWL DL and OWL Full.

ter Horst [2005a,b] later defined pD* semantics as an extension of RDFS semantics to cover a subset of OWL, along with a set of associated pD* entailment rules. Although discernibly similar to DLP, pD*

[3] In fact, it is possible to emulate datatype "keys" using the new OWL 2 `owl:hasKey` primitive, but such an axiom is somewhat unwieldy, and, in some corner cases, does not correspond directly to the semantics of the current FOAF definition for `foaf:mbox_sha1sum`; cf. `http://www.mail-archive.com/public-lod@w3.org/msg06086.html`; retr. 2011/01/11.

includes support for additional constructs such as owl:SymmetricProperty, owl:sameAs, owl:FunctionalProperty, owl:InverseFunctionalProperty, and owl:hasValue (the former, at least, could trivially have been included in DLP)—again, the pD* entailment rules can only offer partial (intensional) support for the OWL semantics of axioms involving, e.g., owl:allValuesFrom. The set of pD* entailment rules—commonly dubbed "OWL Horst"—was subsequently implemented by a number of systems, including OWLIM [Kiryakov et al., 2005], BigOWLIM [Bishop et al., 2011], and more recently, WebPIE [Urbani et al., 2010]. A variant of pD* called OWL Prime [Wu et al., 2008] added partial support for owl:differentFrom and owl:disjointWith; OWL Prime was integrated into the Oracle 11g RDF storage system.

Given adoption of partial rule-support for OWL, Allemang and Hendler [2008] moved away from purely formal considerations and identified criteria important for adoption and uptake of an extension to RDFS: (i) *pedagogism*—additional OWL constructs should be compatible with the underlying intuition behind RDFS; (ii) *practicality*—additional OWL constructs should have practical and tangible applications for modelling; (iii) *computational flexibility*—additional OWL constructs should be implementable using a number of technologies (including rule engines). Based on these criteria and an informal poll of adopters, Allemang and Hendler [2008] defined *RDFS-plus*, which extends RDFS with support for owl:inverseOf, owl:SymmetricProperty, owl:TransitiveProperty, owl:equivalentProperty, owl:equivalentClass, owl:sameAs, owl:FunctionalProperty, owl:InverseFunctionalProperty, owl:DatatypeProperty and owl:ObjectProperty.[4] Thereafter, a number of systems have implemented variations on this theme. The SPIN rule engine has defined a ruleset to support inferencing over the RDFS-plus fragment as originally defined.[5] The AllegroGraph RDF store supports a fragment they call "RDFS++", which extends RDFS with owl:sameAs, owl:inverseOf and owl:TransitiveProperty.[6] The Oracle 11g RDF store defines its own variant of "RDFS++" which extends RDFS with owl:sameAs and owl:InverseFunctionalProperty.[7]

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 [Grau et al., 2009]. 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), and where—taking inspiration from DLP—there is a known relationship between both formal views of the profile: this is provided by Theorem PR1 [Grau et al., 2009] which states that given a valid OWL 2 RL ontology with certain restrictions (no use of metamodelling or extension of annotation properties, etc.), the same *assertional* entailments are given by the Direct Semantics and the provided OWL 2 RL/RDF entailment rules.

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 [Hayes, 2004]. The atoms of these rules comprise primarily of ternary predicates encoding generalised RDF triples; some rules have a special head (denoted false) which indicates that a ground instance of the body of the rule is inconsistent. Given that OWL 2 RL/RDF is currently the most comprehensive, standard means of supporting RDFS *and* OWL entailment using rules—and given that it largely subsumes the entailment possible through RDFS, DLP, pD*, RDFS-Plus, etc.—in this thesis we select OWL 2 RL/RDF as the basis for our reasoning procedure. Appendix B provides a categorisation of the OWL 2 RL/RDF rules according to how they are supported; the full list of OWL 2 RL/RDF rules is also available at [Grau et al., 2009].

[4] We ourselves note that this corresponds to the set of OWL primitives whose axioms are expressible using a single triple, but excluding OWL terms relating to annotations, versioning, and those which can lead to inconsistency (e.g., owl:differentFrom, owl:disjointWith, etc.).

[5] See http://topbraid.org/spin/rdfsplus.html; retr. 2010/11/30

[6] See http://www.franz.com/agraph/support/learning/Overview-of-RDFS++.lhtml; retr. 2010/11/30

[7] See http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28397/owl_concepts.htm; retr. 2010/11/30

5.1.3 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 independence*: 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. In engineering terms, this can be reduced to the simple trade-off of precomputing and storing answers—suitable for frequently accessed data which are expensive to compute but small enough to efficiently store—versus storing a minimal index from which more complex answers can be computed on demand—more suitable where the "answer space" is too large to materialise, where answers can be efficiently computed at runtime, and/or where there is little scope for reuse of computed answers. 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.

5.1.4 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:

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

Note that this rule does not constrain $\mathtt{lt}_1$ or $\mathtt{lt}_2$ 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. Typically, materialisation engines support the above 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.

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; for example, consider the following two triples:

(owl:sameAs, owl:sameAs, rdf:type)
(owl:sameAs, rdfs:domain, rdfs:Resource)

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 (§ 3.1) mentioned in the OWL 2 RL/RDF ruleset and the graph (a.k.a. the Herbrand universe). The process is as follows:

- every term in C is inferred to have the owl:sameAs relation to itself by rule eq-ref;
- thus, by the second triple above and rule prp-dom, every term is inferred to be a member of the class rdfs:Resource with the rdf:type relation;
- thus, by the first triple above and rule eq-rep-p, every term is inferred to have the owl:sameAs relation to rdfs:Resource;
- thus, by the semantics of equality supported by rules eq-*, all terms are pairwise equivalent, and every term is replaceable by every other term in every position of a triple.

Thus, these two triples entail $|C|^3$ triples by the OWL 2 RL/RDF rules, and as such, emulate the explosive nature of inconsistency without actually requiring any inconsistency. (Note that for pD*, the first triple alone is sufficient to give cubic reasoning since everything is already entailed to be a member of rdfs:Resource through rules rdfs4a & rdfs4b—strictly speaking however, pD* does not support "fully generalised" triples, and so triples with literals as subject or predicate will not be inferred.)

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 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. This immediately adds C triples to the materialised graph, where these triples are probably of little interest to the consumer system, can easily be supported by backward-chaining, and typically do not lead to further novel inferences.[8] 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.

5.2 Distinguishing Terminological Data

Given a sufficiently large corpora collected from the Web, the percentage of terminological data (§ 3.5) is relatively small when compared to the volume of assertional data: typically—and as we will see in § 5.4.4—terminological data represent less than one percent of such a corpus [Hogan et al., 2009b, 2010c]. 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.

As such, this approach is related to the area of partial evaluation and program specialisation of Logic Programs [Komorowski, 1982; Lloyd and Shepherdson, 1991; Jones et al., 1993]:

> *"A partial evaluator is an algorithm which, when given a program and some of its input data, produces a so-called residual or specialized program. Running the residual program on the remaining input data will yield the same result as running the original program on all of its input data. [...] The essence of applying partial evaluation is to solve a problem indirectly, using some relatively static information to generate a faster special-purpose program, which is then run on various data supplied more dynamically."*
>
> —**[Jones et al., 1993]**

Herein, 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.

To begin, we formalise the notion of a *T-split rule* which distinguishes between terminological and assertional atoms (T-atoms/A-atoms as defined for RDFS/OWL in § 3.5).

Definition 5.1 (T-split rule) *A* T-split rule *R is given as follows:*

$$H \leftarrow A_1, \ldots, A_n, T_1, \ldots, T_m \quad (n, m \geq 0)\,, \tag{5.1}$$

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

Example 5.1 *Let R_{EX} denote the following rule:*

[8] Where they do lead to novel inferences, we can rewrite the rules involved to implicitly support the tautological triples since they hold true for *every* term. For example, `eq-diff1` requires reflexive `owl:sameAs` statements to detect inconsistencies for reflexive `owl:differentFrom` statements, but we can easily add an additional rule to check specifically for such statements.

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

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 5.2 (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:*

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

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

Example 5.2 *Consider the T-split rule R_{EX} as before:*

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

Now let

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

Here,

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

◊

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 § 3.4 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 5.3 (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 $P^F := \{R \in P \mid \mathsf{Body}(R) = \emptyset\}$ be the set of facts in P,*[9] *let $P^{T\emptyset} := \{R \in P \mid \mathsf{TBody}(R) \neq \emptyset, \mathsf{ABody}(R) = \emptyset\}$, let $P^{\emptyset A} := \{R \in P \mid \mathsf{TBody}(R) = \emptyset, \mathsf{ABody}(R) \neq \emptyset\}$, and let $P^{TA} := \{R \in P \mid \mathsf{TBody}(R) \neq \emptyset, \mathsf{ABody}(R) \neq \emptyset\}$. Note that $P = P^F \cup P^{T\emptyset} \cup P^{\emptyset A} \cup P^{TA}$. Now, let*

$$TP := P^F \cup P^{T\emptyset}$$

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

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*

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

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

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

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* $\mathsf{lm}^T(P)$*.*

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

Proof: Given that, (i) by definition $\mathfrak{T}_P(\mathsf{lm}(P)) = \mathsf{lm}(P)$—i.e., $\mathsf{lm}(P)$ represents a fixpoint of applying the rules, (ii) $\mathfrak{T}_P$ is monotonic, and (iii) $TP \subseteq P \subseteq \mathsf{lm}(P)$, it follows that $\mathsf{lm}(TP) \subseteq \mathsf{lm}(P)$. Analogously, given $AP := \mathsf{lm}(TP) \cup P^{\emptyset A} \cup P^{A+}$, since $P^{\emptyset A} \subseteq P$, $P^{TA} \subseteq P$ and all P^{A+} rules are P^{TA} rules partially ground by $\mathsf{lm}(TP) \subseteq \mathsf{lm}(P)$, then $\mathsf{lm}(AP) \subseteq \mathsf{lm}(P)$. Since $\mathsf{lm}(AP) := \mathsf{lm}^T(P)$, the proposition holds. □

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 5.4 (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*:*

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

(Here, we recall the $\triangleright$ notation of an instance from § 3.4 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 5.2 (T-split conditional completeness) *For any program* P*, its terminological program* TP *and its assertional program* AP*, if* $\mathsf{TBox}(P, \mathsf{lm}(TP)) = \mathsf{TBox}(P, \mathsf{lm}(AP))$*, then* $\mathsf{lm}(P) = \mathsf{lm}^T(P)$*.*

Proof: Given the condition that $\mathsf{TBox}(P, \mathsf{lm}(TP)) = \mathsf{TBox}(P, \mathsf{lm}(AP))$, we can say that $\mathsf{lm}(AP) := \mathsf{lm}(\mathsf{lm}(TP) \cup P^{A+} \cup P^{\emptyset A}) = \mathsf{lm}(\mathsf{lm}(TP) \cup P^{A+} \cup P^{\emptyset A} \cup P^{T\emptyset} \cup P^{TA})$ since by the condition there is no new terminological knowledge to satisfy rules in $P^{T\emptyset} \cup P^{TA}$ after derivation of TP and P^{A+}; this can then be reduced to $\mathsf{lm}(AP) = \mathsf{lm}(\mathsf{lm}(P^F \cup P^{T\emptyset}) \cup P^F \cup P^{\emptyset A} \cup P^{T\emptyset} \cup P^{TA} \cup P^{A+}) = \mathsf{lm}(\mathsf{lm}(TP) \cup P \cup P^{A+}) = \mathsf{lm}(AP \cup P)$. Since by Theorem 5.1 we (unconditionally) know that $\mathsf{lm}(AP) \subseteq \mathsf{lm}(P)$, then we know that $\mathsf{lm}(AP) = \mathsf{lm}(AP \cup P) = \mathsf{lm}(P)$ under the stated condition. □

The intuition behind this proof is that applying the standard program P over the T-split least model $\mathsf{lm}^T(P)$ (a.k.a. $\mathsf{lm}(AP)$) under the condition of completeness cannot give any more inferences, and since the

T-split least model contains all of the facts of the original program (and is sound), it must represent the standard least model. Note that since $\mathsf{lm}^T(P) = \mathsf{lm}(AP)$, the condition for completeness can be rephrased as $\mathsf{TBox}(\mathsf{lm}^T(P)) = \mathsf{TBox}(\mathsf{lm}(TP))$, or $\mathsf{TBox}(\mathsf{lm}^T(P)) = \mathsf{TBox}(\mathsf{lm}(P))$ given the result of the proof.

We now briefly give a corollary which rephrases the completeness condition to state that the T-split least model is complete if assertional rules do not infer terminological facts.

Corollary 5.3 (Rephrased condition for T-split completeness) *For any program P, if a rule with non-empty* ABody *does not infer a terminological fact, then* $\mathsf{lm}(P) = \mathsf{lm}^T(P)$*.*

Proof: It is sufficient to show that $\mathsf{TBox}(P, \mathsf{lm}(TP)) \neq \mathsf{TBox}(P, \mathsf{lm}(AP))$ can only occur if rules with non-empty ABody infer $\mathsf{TBox}(P, \mathsf{lm}(AP)) \setminus \mathsf{TBox}(P, \mathsf{lm}(TP))$. Since (i) $\mathsf{lm}(TP)$ contains all original facts and all inferences possible over these facts by rules with empty ABody, (ii) $\mathsf{lm}(TP) \subseteq AP$, (iii) all proper rules in AP have non-empty ABody, then the only new facts that can arise (terminological or not) after the computation of $\mathsf{lm}(TP)$ are from (proper) rules with non-empty ABody during the computation of $\mathsf{lm}(AP)$.

□

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-* in Table B.7).[10]

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

Definition 5.5 (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*:*

- *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 5.3 *As an example of incompleteness caused by non-standard use of the meta-class* `owl:InverseFunctionalProperty`*, consider:*

1_a.	(`ex:MyKeyProperty`, `rdfs:subClassOf`, `owl:InverseFunctionalProperty`)
2_a.	(`ex:isbn13`, `a`, `ex:MyKeyProperty`)
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`)

[10] 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 [de Bruijn and Heymans, 2007].

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-properties; 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 5.4 (Conditional completeness for standard use) *Let $\mathcal{O2R}'$ 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 $\mathcal{O2R}'$, and (ii)* rdf:type, rdf:first, rdf:list*; and let $P := \mathcal{O2R}' \cup I$; then, it holds that $\mathsf{lm}(P) = \mathsf{lm}^T(P)$.*

Proof: *(Sketch)* First, we note that the set of axiomatic facts in $\mathcal{O2R}'$ (Table B.1) does not contain any non-standard triples.

Second, given that P does not contain any non-standard triples, we need to show that non-standard triples cannot be generated during the computation of $\mathsf{lm}(TP)$, where TP is the terminological program of P (recall: $TP := P^F \cup P^{T\emptyset}$). Note that $P^{T\emptyset}$ corresponds to Table B.2, in which we observe by inspection that non-standard triples can only be created if a restricted term is substituted for a variable in the rule body (i.e., that no atom in any of the heads is trivially *only* grounded by non-standard triples—none of the atoms in the heads contains a restricted constant in a non-standard position), which can only happen if the body is grounded by a non-standard triple since (i) the atoms in the rule bodies are all terminological; *and* (ii) any restricted term substituted for a variable in a body atom must be in a non-standard position of a terminological triple since, for all rule bodies, no variable appears in the predicate position or object position of an rdf:type atom—i.e., all variables in all rule bodies are necessarily in non-standard positions for any restricted term.

Next, we want to show that the assertional program AP (recall $AP := \mathsf{lm}(TP) \cup P^{\emptyset A} \cup P^{A+}$) cannot introduce any new terminological triples if there are no non-standard triples initially in AP. For convenience, we first note that by Theorem 5.1, the T-split least model is necessarily a subset of the standard least model, and so we can inspect the rules in $P^{\emptyset A} \cup P^{TA}$ without considering the T-grounding separately. Now, we can apply an analogous inspection of the rules in Tables B.3, B.4, B.7, B.8, B.9 (ignoring eq-rep-s, eq-rep-p, eq-rep-o by the assumption, and rules with false in the head which cannot infer triples). All such rules can be observed to have atoms in the head of the following generic form:[11]

[11] Again, note that rules with multiple atoms in the head (e.g., $H_1, H_2 \leftarrow B$) serve as a shortcut for multiple rules with single atoms in the head (e.g., $H_1 \leftarrow B$, $H_2 \leftarrow B$)—i.e., that we imply conjunction (e.g., $H_1 \wedge H_2 \leftarrow B$) (§ 3.4).

1. $(?\mathtt{v}', \mathtt{owl:sameAs}, ?\mathtt{v}'')$
2. $(?\mathtt{v}_a, \mathtt{rdf:type}, ?\mathtt{v}_b)$
3. $(?\mathtt{v}_1, ?\mathtt{v}_2, ?\mathtt{v}_3)$

By definition, rules with heads of the form (1) cannot directly infer terminological triples since a triple with owl:sameAs as predicate cannot be terminological; thus we can now ignore all rules in Tables B.7 & B.8. For rules with heads of the form (2), $?\mathtt{v}_b$ must be substituted by a meta-class for the direct inference to be terminological; however, inspecting each such rule, $?\mathtt{v}_b$ always appears in a non-standard position for a terminological atom in the head, and thus $?\mathtt{v}_b$ can only be substituted by a meta-class if the atom is grounded by a non-standard triple. For rules with heads of the form (3), a grounding can only be terminological if $?\mathtt{v}_2$ is substituted by rdf:type, rdf:first, rdf:rest or a meta-property: again, by inspection, grounding this variable in each rule always leaves a terminological atom that can only be grounded by a non-standard triple. Thus, we know that new terminological triples cannot be directly inferred in the absence of non-standard triples. Next, note that non-standard triples are terminological, so $\mathfrak{T}_{AP}$ also cannot introduce non-standard triples. Again, by induction, we have that if AP does not contain non-standard triples, then $\mathsf{lm}(AP)$ does not contain non-standard triples, and $\mathsf{TBox}(P, \mathsf{lm}(TP)) = \mathsf{TBox}(P, \mathsf{lm}(AP))$ (recall $\mathsf{lm}(TP) \subseteq AP$). We've reached the condition for completeness from Theorem 5.2, and thus $\mathsf{lm}(P) = \mathsf{lm}^T(P)$ and the theorem holds.

□

Briefly, we note that Weaver and Hendler [2009] 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 5.2. 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 now take some examples:

Example 5.4 *As an example of how new terminology can be inferred from rules with assertional atoms through the semantics of replacement for* owl:sameAs*, consider:*

1_a. (ex:isbn13, a, owl:InverseFunctionalProperty)
2_a. (ex:isbnThirteen, owl:sameAs, ex:isbn13)
3_a. (ex:The_Road, ex:isbn13, "978-0307265432")
4_a. (ex:Road%2C_The, ex:isbnThirteen, "978-0307265432")

where by rule eq-rep-s *and triples* (1_a) *and* (2_a)*, we infer that:*

5_a. (ex:isbnThirteen, a, owl:InverseFunctionalProperty)

which is a terminological triple inferred through the assertional rule eq-rep-s*. However, by rule* eq-rep-p *and triples* (2_a)*,* (3_a) *and* (4_a)*, we infer that:*

6_a. (ex:The_Road, ex:isbnThirteen, "978-0307265432")
7_a. (ex:Road%2C_The, ex:isbn13, "978-0307265432")

Now, note that the inferences:

8_a. (ex:The_Road, owl:sameAs, ex:Road%2C_The)
9_a. (ex:Road%2C_The, owl:sameAs, ex:The_Road)

hold through rule prp-ifp *and triples* (1_a)*,* (3_a) *and* (7_a)*—the terminological triple* (5_a) *is redundant since* *by* `eq-rep-p`, *all of the relevant relations will be expressed using* both *predicates* `ex:isbnThirteen` *and* `ex:isbn13`, *and the inverse-functional inferences will be given by the original terminological triple* (1_a) *for the triples using the latter predicate. Thus, in this example,* completeness is not affected.

Typically, incompleteness can only be caused when `owl:sameAs` *relations involve the restricted vocabulary itself. One can consider again Example 5.3 replacing the predicate of triple* (1_a) *with* `owl:sameAs`.

However, there are some other corner cases that can force incompleteness. To see this, we need to look at another (rather unappealing) example. Consider:[12]

1_b. || (`ex:isbn13`, `rdfs:domain`, `_:card`)
2_b. || (`_:card`, `owl:maxCardinality`, `ex:one`)
3_b. || (`_:card`, `owl:onProperty`, `ex:isbn13`)
4_b. || (`ex:one`, `owl:sameAs`, `1`)
5_b. || (`ex:The_Road`, `ex:isbn13`, `"978-0307265432"`)
6_b. || (`ex:Road%2C_The`, `ex:isbn13`, `"978-0307265432"`)

By rule `prp-dom` *and triples* (1_b), (5_b) *and* (6_b), *we have:*

7_b. || (`ex:The_Road`, `a`, `_:card`)
8_b. || (`ex:Road%2C_The`, `a`, `_:card`)

Now, by rule `eq-rep-o`, *we have:*

9_b. || (`_:card`, `owl:maxCardinality`, `1`)

which is a new terminological triple inferred by an assertional rule, and one which is part of the OWL 2 RL/RDF T-Box. Now, by rule `cax-maxc2` *and triples* (3_b), (5_b), (6_b), (7_b), (8_b), (9_b), *we should infer:*

10_b. || (`ex:The_Road`, `owl:sameAs`, `ex:Road%2C_The`)
11_b. || (`ex:Road%2C_The`, `owl:sameAs`, `ex:The_Road`)

but we miss these inferences since (9_b) *is not in the terminological program. By a similar principle,* `owl:sameAs` *relations affecting* `rdf:nil` *can also cause incompleteness.*

Finally, incompleteness can also occur if `owl:sameAs` *causes alignment of a terminological join; consider the above example again, but replace* (2_b) *with:*

$\overline{2}_b$. || (`_:cardx`, `owl:maxCardinality`, `ex:one`)

and add:

x_b. || (`_:cardx`, `owl:sameAs`, `_:card`)

such that we should be able to infer the original (2_b) *and follow the same inference, but inferring* (2_b) *requires the application of* `eq-rep-s` *over triples* ($\overline{2}_b$) *and* (x_b) *in the assertional program, and thus* (2_b) *will not be considered in the terminological program.* ◊

Although we could attempt to broaden the scope of non-standard usage to tackle such problematic `owl:sameAs` relation(s), we note that such `owl:sameAs` triples can also be *inferred*—i.e., in order to extend the scope of non-standard use in a manner which facilitates a completeness guarantee in the presence of `eq-rep-*` rules, it is not enough to only control the use of explicit `owl:sameAs`, but also the various primitives which can infer `owl:sameAs` for the problematic terms. This would lead to either (i) a very broad generic

[12] Note that for the term `1`, we use Turtle syntax and imply that it matches any literal which shares the value space of `"1"^^xsd:decimal` —i.e., that the semantics of rule `dt-eq` are implied in the shortcut syntax.

restriction—for example, disallowing use of `owl:sameAs`, as well as (inverse-)functional-properties, keys and cardinalities-of-one; or (ii) a narrower but very intricate restriction, which blocks the various (indirect) inference paths for only those problematic `owl:sameAs` relations. For (i), such restrictions would cover a lot of real-world data; for (ii), it seems that the myriad (indirect) inference paths for `owl:sameAs` relations would lead to very nebulous restrictions.

Having sketched the nature of the issue, we leave it open: 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 § 5.4. Instead, we look at partially supporting the semantics of equality by non-standard means in Chapter 7. In any case, we believe that in practice, T-split incompleteness through such `owl:sameAs` relations would only occur for rare corner cases.

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.

5.2.1 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 are partially-grounded by these terminological 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., [Ullman, 1989]).

We give a more formal break-down of the application of the assertional program in Algorithm 5.1. 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 [Golbreich and Wallace, 2009].

Algorithm 5.1: Reason over the A-Box

```
Require: ABOX: A                                                   /* $\{t_0 \ldots t_m\}$ */
Require: ASSERTIONAL PROGRAM: $AP$                    /* $\{R_0 \ldots R_n\}, \mathsf{TBody}(R_i) = \emptyset$ */
 1: Index := {}                                                    /* triple index */
 2: LRU := {}                                /* fixed-size, least recently used cache */
 3: for all $t \in$ A do
 4:   $G_0 := \{\}, G_1 := \{t\}, i := 1$
 5:   while $G_i \neq G_{i-1}$ do
 6:     for all $t_\delta \in G_i \setminus G_{i-1}$ do
 7:       if $t_\delta \notin$ LRU then              /* if $t_\delta \in$ LRU, make $t_\delta$ most recent entry */
 8:         add $t_\delta$ to LRU                        /* remove eldest entry if necessary */
 9:         output($t_\delta$)
10:         for all $R \in AP$ do
11:           if $|\mathsf{Body}(R)| = 1$ then
12:             if $\exists\theta$ s.t. $\{t_\delta\} = \mathsf{Body}(R)\theta$ then
13:               $G_{i+1} := G_{i+1} \cup \mathsf{Head}(R)\theta$
14:             end if
15:           else
16:             if $\exists\theta$ s.t. $t_\delta \in \mathsf{Body}(R)\theta$ then
17:               if $t_\delta \notin$ Index then
18:                 Index := Index $\cup \{t_\delta\}$
19:                 for all $\theta$ s.t. $\mathsf{Body}(R)\theta \subseteq$ Index$, t_\delta \in \mathsf{Body}(R)\theta$ do
20:                   $G_{i+1} := G_{i+1} \cup \mathsf{Head}(R)\theta$
21:                 end for
22:               end if
23:             end if
24:           end if
25:         end for
26:       end if
27:     end for
28:     $i$++
29:     $G_{i+1} :=$ copy($G_i$)                 /* copy inferences to new set to avoid cycles */
30:   end while
31: end for
32: return output                                              /* on-disk inferences */
```

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 triples which are matched by a rule with a multi-atom body (Lines 15–24). For indexed triples, aside from the LRU cache, we can additionally check to see if that triple has been indexed before (Line 17) and we can apply a semi-naïve check to ensure that we only materialise inferences which involve the current triple (Line 19). 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**:

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

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 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 § 5.4.1, but first we look at some generic optimisations for the assertional program.

5.3 Optimising the Assertional Program

Note that in Algorithm 5.1 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.

5.3.1 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 through a *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 5.6 (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 5.7 (Equivalent atoms) *Two atoms are considered* equivalent *(denoted $A_1 \mathrel{\lhd\!\rhd} A_2$ reflecting the fact that both atoms are instances of each other) iff they are unifiable by a bijective variable rewriting:*[13]

$$A_1 \mathrel{\lhd\!\rhd} A_2 \Leftrightarrow \exists \nu \text{ s.t. } A_1\nu = A_2$$

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

$$R_1 \mathrel{\lhd\!\rhd}_{\mathrm{b}} R_2 \Leftrightarrow \mathsf{Body}(R_1) \mathrel{\lhd\!\rhd} \mathsf{Body}(R_2) \Leftrightarrow \exists \nu \text{ s.t. } \mathsf{Body}(R_1)\nu = \mathsf{Body}(R_2)$$

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

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

[13] 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.

We now briefly demonstrate that our equivalence relations meet the necessary requirements to be called such:

Proposition 5.5 *The equivalence relations $\lhd\!\rhd$, $\lhd\!\rhd_{\mathrm{b}}$ and $\lhd\!\rhd_{\mathrm{r}}$ are reflexive, symmetric and transitive.*

Proof: For $\lhd\!\rhd$, the proposition follows trivially from the fact that ν is automorphic. Reflexivity is given by the necessary existence of the identity morphism $\nu(A) = A$ which gives $A \lhd\!\rhd A$. Symmetry is given by the necessary existence of the inverse (auto)morphism ν^{-1} where:

$$A_1 \lhd\!\rhd A_2 \Leftrightarrow \exists\nu \text{ s.t. } (A_1\nu = A_2) \Leftrightarrow \exists\nu^{-1} \text{ s.t. } (A_2\nu^{-1} = A_1) \Leftrightarrow A_2 \lhd\!\rhd A_1$$

Transitivity is given by the necessary existence of the composite (auto)morphism $\nu_x \circ \nu_y$ where:

$$\begin{gathered} A_1 \lhd\!\rhd A_2 \lhd\!\rhd A_3 \Leftrightarrow \exists\nu_x, \nu_y \text{ s.t. } (A_1\nu_x = A_2 \wedge A_2\nu_y = A_3) \\ \Leftrightarrow \exists\nu_x \circ \nu_y \text{ s.t. } (A_1\nu_x \circ \nu_y = A_3) \Leftrightarrow A_1 \lhd\!\rhd A_3 \end{gathered}$$

Thus, the proposition holds for $\lhd\!\rhd$, and can be proven analogously for $\lhd\!\rhd_{\mathrm{b}}$ and $\lhd\!\rhd_{\mathrm{r}}$. □

Using these equivalence relations, we can now define our *rule-merge* function (again recall from § 3.4 our interpretation of multi-atom heads as being conjunctive, and a convenient representation of the equivalent set of rules):

Definition 5.8 (Rule merging) *Given an equivalence class of rules $[R]_{\lhd\!\rhd_{\mathrm{b}}}$—a set of rules between which $\lhd\!\rhd_{\mathrm{b}}$ holds—select a* canonical rule $R \in [R]_{\lhd\!\rhd_{\mathrm{b}}}$*; we can now describe the* rule-merge *of the equivalence class as*

$$\mathsf{merge}([R]_{\lhd\!\rhd_{\mathrm{b}}}) := \mathsf{Head}_{[R]_{\lhd\!\rhd_{\mathrm{b}}}} \leftarrow \mathsf{Body}(R)$$

where

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

Now take a program P and let:

$$P/\lhd\!\rhd_{\mathrm{b}} := \{[R]_{\lhd\!\rhd_{\mathrm{b}}} \mid R \in P\}$$

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

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

Example 5.5 *Take three T-ground rules:*

(?x, a, foaf:Person) ← (?x, foaf:img, ?y)
(?s, foaf:depicts, ?o) ← (?s, foaf:img, ?o)
(?a, foaf:depicts, ?b) ← (?a, foaf:img, ?b)

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

(?x, a, foaf:Person),(?x, foaf:depicts, ?y) ← (?x, foaf:img, ?y)

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

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

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

Proof: *(Sketch)* First, we can show that $\mathfrak{T}_P = \mathfrak{T}_{\mathsf{merge}(P)}$ since (i) the bijective variable rewriting does not affect rule application; (ii) concatenation of rules with equivalent bodies does not affect the immediate consequence operator over that set of rules. Give that the immediate consequences of the program are the same, proof that the least models are the same can be demonstrated by induction over natural numbers. □

5.3.2 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 5.9 (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:*

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

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 5.6 *Given a triple:*

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

and a simple example ruleset:

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

$\mathsf{lookup}(t, P)$ *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., $\{$(`?x`, `a`, `foaf:Person`)$\}$ is put into the $(?,p,o)$ index with ($\mathtt{a}$, $\mathtt{foaf:Person}$) 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., [Ramakrishnan et al., 1990]):

Definition 5.10 (Rule graph) *A* rule graph *is defined as a directed graph:*

$$\Gamma := (P, \hookrightarrow)$$

such that:[14]

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

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 5.11 (Rule-graph labelling) *Let Λ denote a labelling function as follows:*

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

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

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

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 5.7 *Take the two rules:*

$$\begin{array}{r|c} R_i. & (\mathtt{y},\ \mathtt{a},\ \mathtt{foaf{:}Image}),(\mathtt{?x},\ \mathtt{a},\ \mathtt{foaf{:}Person}) \leftarrow (\mathtt{?x},\ \mathtt{foaf{:}img},\ \mathtt{?y}) \\ R_j. & (\mathtt{s},\ \mathtt{a},\ \mathtt{foaf{:}Agent}) \leftarrow (\mathtt{?s},\ \mathtt{a},\ \mathtt{foaf{:}Person}) \end{array}$$

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

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.

5.3.3 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

[14] Here, we recall from § 3.4 the '$\triangleright$' notation for an instance.

strong dependencies, where the successful application of one rule *will* always lead to the successful application of another. In such cases, 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 5.8 *Take rules*

$$\begin{array}{r|c}
R_i. & \texttt{(?x, a, foaf:Person),(?y, a, foaf:Image)} \leftarrow \texttt{(?x, foaf:img, ?y)} \\
R_j. & \texttt{(?s, a, foaf:Agent)} \leftarrow \texttt{(?s, a, foaf:Person)} \\
R_k. & \texttt{(?y, a, rdfs:Class)} \leftarrow \texttt{(?x, a, ?y)}
\end{array}$$

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:

$$\begin{array}{r|c}
R'_i. & \texttt{(?x, a, foaf:Person),(?y, a, foaf:Image),(?x, a, foaf:Agent),} \\
 & \texttt{(foaf:Person, a, rdfs:Class),(foaf:Image, a, rdfs:Class),} \\
 & \texttt{(foaf:Agent, a, rdfs:Class)} \leftarrow \texttt{(?x, foaf:img, ?y)}
\end{array}$$

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

$$\begin{array}{r|c}
R'_j. & \texttt{(?s, a, foaf:Agent),(foaf:Agent, a, rdfs:Class)} \leftarrow \texttt{(?s, a, foaf:Person)}
\end{array}$$

Thus, the index now stores R'_i, R'_j, R_k, but without the links between them. (Note that R'_j does not contain the atom (`foaf:Person`, `a`, `rdfs:Class`) *in its head since R'_k will also be applied on the input triple, although not on the consequences of R'_j.)* ◊

However, as we will see in § 5.3.4, 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:

$$\begin{array}{r|c}
1. & \texttt{(ex:Fred, a, foaf:Person)} \\
2. & \texttt{(ex:Fred, foaf:img, ex:FredsPic)}
\end{array}$$

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`).

5.3.4 Preliminary Performance Evaluation

We now perform some (relatively) small-scale experiments to empirically (in)validate our optimisations for the assertional program execution.

We applied reasoning for RDFS (minus the infinite `rdf:_n` axiomatic triples [Hayes, 2004]), pD* and OWL 2 RL/RDF over LUBM(10) [Guo et al., 2005], consisting of about 1.27 million assertional triples and 295 terminological triples.[15] 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;
2. Nl: **no partial evaluation with linked (meta-)rule index**;
3. P: **partial-evaluation**: generating and applying an assertional program;

[15] Note that we exclude lg/gl rules for RDFS/pD* since we allow generalised triples [Grau et al., 2009]. We also restrict OWL 2 RL/RDF datatype reasoning to apply only to literals in the 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 5.1 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 5.1).

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 5.1: 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)

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× to 4× 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 for the 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× more duplicates, negating the benefits of fewer rule applications. Similarly, for OWL 2 RL/RDF, we encountered a 4% increase in runtime over 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).[16]

5.4 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:

- first describe, motivate and characterise the scalable subset of OWL 2 RL/RDF that we implement (§ 5.4.1) based partially on the discussion in the previous section;
- introduce and describe *authoritative reasoning*, whereby we include cautious consideration of the source of terminology into the reasoning process (§ 5.4.2);
- outline our distribution strategy for reasoning (§ 5.4.3);

[16] 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.

- evaluate our methods (§ 5.4.4) by applying reasoning over the corpus crawled in the previous chapter.

5.4.1 "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 § 5.1.4 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 [Cosmadakis et al., 1988]).[17] 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 [Cosmadakis et al., 1988]. 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.

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 5.12 (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 B.1–B.4 (Appendix B).

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 5.7 (Assertional partitionability) *Let I be any interpretation, and $\{I_1, \ldots, I_n\}$ be any*

[17] There is no relation between a linear program in our case, and the field of Linear Programming [Vanderbei, 2008].

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: 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 5.8 (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:

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

Proof: The proposition breaks the least model into two parts: The first part consists of τ^3 triples representing the cardinality of the set of all possible triples that can be generated from the set of constants in the heads of the proper rules—clearly, no more triples can be generated without accessing the Herbrand universe of the assertional facts. The second part of the least model consists of $\alpha(9\tau^2 + 27\tau + 27)$ triples generated from the assertional facts. From Proposition 5.7, we know that the least model can be viewed as the set union of the consequences from each individual triple. For each triple, the program has access to the τ terms in the Herbrand universe of the proper-rule heads, and three additional terms from the triple itself; the total number of unique possible triples from this extended Herbrand universe is:

$$(\tau + 3)^3 = \tau^3 + 9\tau^2 + 27\tau + 27$$

However, we have already counted the τ^3 triples that can be created purely from the former Herbrand universe, and thus the total number of unique triples that can be derived thereafter comes to:

$$9\tau^2 + 27\tau + 27$$

denoting the number of possible triples which include at least one term from the input triple. Thus, multiplying the total number of triples α, we end up with the maximum total size of the least model given in the proposition. □

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 B, 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 (...for now: we will examine use-cases for these rules in the next two chapters);
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).

5.4.2 Authoritative Reasoning

In preliminary evaluation of our Linked Data reasoning [Hogan et al., 2009b], 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[18] 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** [Grau et al., 2009, Table 6] over any member of `owl:Thing` would infer 55 additional memberships for these obscure union classes. We found another document[19] 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 § 5.4.4).

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 § 5.1.4), 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

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

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

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.

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 § 3.3, 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 5.13 (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:*[20]

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

Thus, a Web source is authoritative for URIs which redirect to it and the blank nodes contained in its associated graph; for example, the FOAF vocabulary is authoritative for terms in its namespace since it follows best-practices and makes its class/property URIs dereference 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 5.2): in particular, we only require amendment to rules where both $\mathsf{TBody}(R) \neq \emptyset$ and $\mathsf{ABody}(R) \neq \emptyset$.

Definition 5.14 (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:*

$$\widehat{\mathsf{Ground}}^T(P, G, s) := \bigcup_{R \in P} \mathsf{Ground}^T(\{R\}, G) \text{ s.t. if } R \in P^{TA} \text{ then } \exists v \in \mathsf{TAVars}(R) \text{ s.t. } \theta(v) \in \mathsf{auth}(s)$$

here recalling the $\mathsf{Ground}^T(P, G)$ *notation for T-ground rule instances from Definition 5.2, and reusing the previous convention that* $P^{TA} := \{R \in P \mid \mathsf{TBody}(R) \neq \emptyset, \mathsf{ABody}(R) \neq \emptyset\}$*.*

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 5.9 (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*

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

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

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:

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

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

Proof: First, we note that P' contains *proper rules* generated from $P^{\emptyset A}$ and P^{TA}—the $\widehat{\mathsf{Ground}}^T$ function will ground any rules in $P^{T\emptyset}$, where the resulting facts will, by definition, not be included in P'. Note that with respect to $P^{\emptyset A}$, by definition the $\widehat{\mathsf{Ground}}^T$ function will not alter these rules such that $\widehat{\mathsf{Ground}}^T(P^{\emptyset A}, G', s') = P^{\emptyset A}$. Thus, we have that

$$P \setminus P' = \widehat{\mathsf{Ground}}^T(P^{TA}, G', s')\,.$$

Letting $P'' := P \setminus P'$, we are now left to prove $\mathsf{lm}(P \cup G) = \mathsf{lm}(P \cup P'' \cup G)$: since our rules are monotonic, the $\subseteq$ inclusion is trivial, where we are left to prove the $\supseteq$ inclusion.

From Definition 5.9, since for all $R^{TA} \in P^{TA}$ there must exist a substitution $v \in \mathsf{TAVars}(R)$ s.t. $\theta(v) \in \mathsf{auth}(s')$, and since v must appear in $\mathsf{ABody}(R^{TA})$, then we know that for all $R'' \in P''$, there exists a constant in R'' for which s' is authoritative; i.e.:

$$\forall R'' \in P'' \text{ it holds that } \mathsf{Const}(\{\mathsf{Body}(R'')\}) \cap \mathsf{auth}(s') \neq \emptyset\,.$$

Now, given the theorem's assumption that $\mathsf{Const}(P \cup G) \cap \mathsf{auth}(s') = \emptyset$, we know that

$$\forall R'' \in P'' \text{ it holds that } \mathsf{Const}(\{\mathsf{Body}(R'')\}) \not\subseteq \mathsf{Const}(P \cup G)\,,$$

and since lm cannot introduce new terms not in the original Herbrand universe, it follows that

$$\forall R'' \in P'' \text{ it holds that } \mathsf{Const}(\{\mathsf{Body}(R'')\}) \not\subseteq \mathsf{Const}\big(\mathsf{lm}(P \cup G)\big)\,.$$

Now, since all R'' are proper rules, it follows that no such R'' can have its body instantiated by $P \cup G$ or $\mathsf{lm}(P \cup G)$, or give an inference therefrom. Finally, by straightforward induction for $\mathfrak{T}_{P \cup P'' \cup G}$, we have that $\mathsf{lm}(P \cup G) = \mathsf{lm}(P \cup P'' \cup G)$. Hence the proposition holds. □

Corollary 5.10 *Given the same assumption(s) as Theorem 5.4.2, it also holds that* $\mathsf{lm}^T(P \cup G) = \mathsf{lm}^T(P \cup P' \cup G)$.

Proof: Follows from Theorem 5.4.2 by replacing P with its terminological program TP and its assertional program AP. □

Example 5.9 *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:

$$\begin{aligned} G_{EX} := \{\ &(\texttt{foaf:Person, rdfs:subClassOf, foaf:Agent}),\\ &(\texttt{foaf:Agent, rdfs:subClassOf, dc:Agent})\ \} \end{aligned}$$

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 dereferenced by the FOAF terms—i.e., the FOAF vocabulary. Note, for example, 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 Table B.4 (Appendix B).

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 [Motik et al., 2009c] 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 [Grau et al., 2009], which allow for deriving the same inferences on a purely assertional level. Along these lines, in Appendix B, Table B.5, 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—this is because we do not support the respective assertional rules cls-svf1 and cls-avf which both contain two A-atoms, whereas we do support inferencing over `rdfs:subClassOf` through rule cax-sco.[21]

5.4.3 Distributed Reasoning

As previously mentioned, Proposition 5.7 lends itself to a straightforward distribution strategy for applying our A-linear OWL 2 RL/RDF subset. Given that the corpus is pre-distributed over the machines as a direct result of the crawl (cf. § 4.2), we first 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:

[21] Informally, we conjecture that such incompleteness would be extremely rare in practice (cf. Table 5.2).

1. **run/gather:** identify and separate out the T-Box from the main corpus in parallel on the slave machines, and subsequently merge the T-Box segments on the master machine;

2. **run:** generate axiomatic triples from the meta-program and apply T-atom only rules on the master machine; subsequently ground the T-atoms in rules with non-zero A-atoms generating the assertional program; optimise the assertional program by merging rules and building a linked rule index;

3. **flood/run:** send the assertional linked rule index to all slave machines, and reason over the main corpus in parallel on each machine.

Note that during the first step, assuming that the data are grouped by context—as is the direct result of our document-by-document crawl—we can detect and remove non-terminological collection triples based on whether or not they form part of an `owl:unionOf` or `owl:intersectionOf` axiom within the current document. If the data are not grouped by context, this step can be performed on the master machine: in our corpus, we found 22,360 `rdf:first` and 22,646 `rdf:rest` triples, which—along with 4,168 `owl:unionOf` triples and 136 `owl:intersectionOf` triples—are processable in sub-second time.

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.

5.4.4 Linked Data Reasoning Evaluation

We now give evaluation of applying our subset of OWL 2 RL/RDF over the 1.118b quads (947m unique triples) of Linked Data crawled in the previous chapter. Note that we also require information about redirects encountered in the crawl to reconstruct the redirs function required for authoritative reasoning (see Definition 5.13) and that we output a flat file of G-Zipped triples.

Survey of Terminology

In this section, we analyse the terminology given by our corpus, including for OWL 2 RL/RDF rules which we do not support (or support in later chapters). By extension, we provide insights into which RDFS and OWL constructs feature prominently in Linked Data vocabularies. Note that for the purposes of this analysis, we only consider duplicate terminological axioms once (and as served in the highest-ranked document).

To begin, in Table 5.2 we present the counts of T-ground rules generated for each OWL 2 RL/RDF rule which handles reasoning over assertional data; we include the primary RDFS/OWL meta-class or meta-property supported by the rule as a mnemomic, where the complete rules are enumerated in Appendix B. Loosely, the count of T-ground rules corresponds with the number of such axioms in the data, with the exception of some "disjunctive" rules involving RDF lists in the body or head where we count fixed length atomic rules as follows:

- for cls-uni, an `owl:unionOf` expression with x constituent classes would count as x rules which infer membership of the union class;

- similarly, for cls-int2, an `owl:intersectionOf` expression with x constituent classes would count as x rules which infer membership of those classes from membership of the intersection class;

- for cax-adc and prp-adp, an `owl:AllDisjointClasses` or `owl:AllDisjointProperties` expression (respectively) with x members would lead to $\frac{x^2-1}{2}$ non-reflexive, pairwise disjointness rules.

Other axioms involving lists—viz., cls-int1/`owl:intersectionOf`, prp-key/`owl:hasKey` and prp-spo2/`owl:-`

#	Rule ID	Sup.?	Meta-Class/-Property	T-Ground Rules			Documents		
				all	auth	¬auth	all	auth	¬auth
1	cax-sco	✓	`rdfs:subClassOf`	307,121	244,608	68,246	51,597	51,034	707
2	cax-eqc1	✓	`owl:equivalentClass`	22,922	22,898	17	22,657	22,652	9
	cax-eqc2	✓	`owl:equivalentClass`	22,922	18,544	4,378	22,657	18,311	4,349
3	prp-dom	✓	`rdfs:domain`	16,204	14,066	2,499	623	609	101
4	prp-rng	✓	`rdfs:range`	13,938	13,662	553	717	704	30
5	cls-uni	✓	`owl:unionOf`	13,004	10,693	2,272	109	98	88
6	prp-spo1	✓	`rdfs:subPropertyOf`	9,109	8,790	361	227	213	24
7	prp-inv1	✓	`owl:inverseOf`	992	746	244	98	93	13
	prp-inv2	✓	`owl:inverseOf`	992	733	256	98	90	19
8	cax-dw	⊥	`owl:disjointWith`	917	714	46	60	56	4
9	prp-fp	C	`owl:FunctionalProperty`	496	445	48	63	59	9
10	cls-svf1	X	`owl:someValuesFrom`	465	420	29	48	45	6
11	cls-int2	✓	`owl:intersectionOf`	325	325	–	12	12	–
12	cls-avf	X	`owl:allValuesFrom`	256	256	–	33	33	–
13	cls-maxc2	C	`owl:maxCardinality`	159	159	–	14	14	–
14	prp-eqp1	✓	`owl:equivalentProperty`	159	121	15	89	87	6
	prp-eqp2	✓	`owl:equivalentProperty`	159	109	39	89	76	16
15	prp-trp	X	`owl:TransitiveProperty`	140	127	8	32	30	2
16	cls-int1	X	`owl:intersectionOf`	136	136	–	12	12	–
17	prp-symp	✓	`owl:SymmetricProperty`	128	108	19	24	22	4
18	cax-adc	⊥	`owl:AllDisjointClasses`	116	110	6	3	3	1
19	prp-ifp	C	`owl:InverseFunctionalProperty`	62	60	12	27	25	4
20	cls-hv1	✓	`owl:hasValue`	13	13	–	6	6	–
	cls-hv2	✓	`owl:hasValue`	13	11	–	6	5	–
21	prp-irp	⊥	`owl:IrreflexiveProperty`	10	10	3	1	1	1
22	prp-asyp	⊥	`owl:AsymmetricProperty`	9	9	3	1	1	1
23	cls-com	⊥	`owl:complementOf`	7	7	–	2	2	–
24	prp-spo2	X	`owl:propertyChainAxiom`	6	6	–	1	1	–
25	cls-svf2	X	`owl:someValuesFrom`	2	2	–	2	2	–
26	prp-key	$\rlap{C}X$	`owl:hasKey`	1	1	–	1	1	–
27	cls-maxc1	⊥	`owl:maxCardinality`	–	–	–	–	–	–
–	cls-maxqc1	⊥	`owl:maxQualifiedCardinality`	–	–	–	–	–	–
–	cls-maxqc2	⊥	`owl:maxQualifiedCardinality`	–	–	–	–	–	–
–	cls-maxqc3	$\rlap{C}X$	`owl:maxQualifiedCardinality`	–	–	–	–	–	–
–	cls-macqc4	C	`owl:maxQualifiedCardinality`	–	–	–	–	–	–
–	prp-adp	⊥	`owl:AllDisjointProperties`	–	–	–	–	–	–
–	prp-pdw	⊥	`owl:propertyDisjointWith`	–	–	–	–	–	–
all rules				410,783	337,889	79,054	56,710	56,139	5,181
rules supported (✓)				408,003	335,429	78,899	56,703	56,132	5,179
rules not supported (X/⊥/C/$\rlap{C}X$)				2,780	2,460	155	160	156	21

Table 5.2: Counts of T-ground OWL 2 RL/RDF rules containing non-empty TBody and ABody from our corpus and count of documents serving the respective axioms

`propertyChainAxiom`—compile into rules with conjunctive bodies, and are thus necessarily expressed in the statistics as a single axiom.

Acknowledging that a single document may be responsible for many such axioms, in Table 5.2 we also present the counts of documents providing each type of axiom; in total, we found terminological data in 86,179 documents, of which 65,861 documents (76.4%) provided terminological axioms required by those rules in Table 5.2,[22] and 56,710 documents (65.8%) provided axioms not already given by a higher ranked document. We note that there are two orders of magnitude more documents defining `rdfs:subClassOf` and `owl:equivalentClass` axioms than any other form of axiom. With respect to documents using `rdfs:subClassOf`, we found (i) 25,106 documents from the `zitgist.com` domain, (ii) 18,288 documents from the `bio2rdf.org` (iii) 5,832 documents from the `skipforward.net` domain, and (iv) 1,016 from the `dbpedia.org` domain; these four publishers account for 97.2% of the documents with subclass axioms. With respect to

[22] Many documents only contained memberships of `owl:Class`, which although terminological, are not required by the rules in the table.

#	Pay-Level-Domain	Axioms
1	`ebusiness-unibw.org`	112,344
2	`ontologydesignpatterns.org`	82,709
3	`dbpedia.org`	78,006
4	`bio2rdf.org`	73,153
5	`zitgist.com`	25,252
6	`umbel.org`	8,680
7	`skipforward.net`	7,695
8	`ontologyportal.org`	7,166
9	`w3.org`	2,822
10	`fao.org`	1,989

Table 5.3: Top ten largest providers of terminological axioms

#	Pay-Level-Domain	Documents
1	`zitgist.com`	25,169
2	`bio2rdf.org`	18,354
3	`skipforward.net`	5,852
4	`umbel.org`	4,330
5	`dbpedia.org`	1,280
6	`uriburner.com`	673
7	`uniprot.org`	364
8	`vocab.org`	105
9	`data.gov.uk`	95
10	`kit.edu`	56

Table 5.4: Top ten largest providers of terminological documents

using `owl:equivalentClass`, we found (i) 18,284 documents again from the `bio2rdf.org` domain, and (ii) 4,330 documents from the `umbel.org` domain; these two publishers account for 99.8% of documents with equivalent-class axioms.

More generally, we found 81 pay-level domains providing terminological data. Table 5.3 enumerates the top ten such domains with respect to the number of *axioms* (more precisely, T-ground rules) they provide, and Table 5.4 enumerates the top ten domains with respect to the number of *documents* containing terminological data. We note that the terminology provided by `ebusiness-unibw.org` is contained within one document,[23] which represents the largest terminological document we encountered.

In Table 5.2, we additionally denote rules that the work in this chapter supports (✓), rules involving inconsistency detection used later in Chapters 6 & 7 (⊥), rules which infer `owl:sameAs` which we support/don't support in Chapter 7 (C/C̸), and remaining rules which we do not support (✗). We observe that our A-linear rules support the top seven most popular forms of terminological axioms in our data (amongst others), and that they support 99.3% of the total T-ground rules generated in our corpus; the 2,780 not supported come from 160 documents spanning 36 domains: 826 come from the `geospecies.org` domain, 318 from the `fao.org` domain, 304 come from the `ontologyportal.org` domain, and so forth.

In Table 5.2, we also give a breakdown of counts for authoritative and non-authoritative T-ground rules and the documents providing the respective axioms. Note that the T-ground rules generated by equivalent authoritative and non-authoritative axioms are counted in both categories, and likewise for a document serving authoritative and non-authoritative axioms of a given type.[24] We find that 82.3% of all generated T-ground rules have an authoritative version, that 9.1% of the documents serve *some* non-authoritative axioms, and that 99% of the documents contain *some* authoritative axioms. We note that:

1. the domain `ontologydesignpatterns.org` publishes 61,887 (78.3%) of the non-authoritative axioms, almost all in one document[25] and almost all of which pertain to rule cax-sco (subclass axioms with a non-authoritative subject);

2. `skipforward.net` publishes a further 5,633 (7.1%) in nine documents, all of which again pertain to cax-sco;

3. `umbel.net` publishes 4,340 such axioms (5.5%) in as many documents, all of which pertain to rule cax-eqc2 (equivalent-class axiom with non-authoritative object);

[23] `http://www.ebusiness-unibw.org/ontologies/eclass/5.1.4/eclass_514en.owl`; retr. 2011/01/06

[24] Thus, the difference between ¬**auth** and (**all** − **auth**) is given by authoritative axioms "echoed" in non-authoritative documents, and documents which provide a mix of authoritative and non-authoritative axioms respectively.

[25] `http://ontologydesignpatterns.org/ont/own/own16.owl`; retr. 2011/01/06

4. `bio2rdf.org` publishes 3,765 such axioms (4.8%) in sixty-seven documents—in particular, 2,144 of these pertain to rule cls-uni (non-authoritative member of a local union class), and 1,564 pertain to rule prp-dom (domain axioms with non-authoritative subject);

5. `fao.org` publishes 1,005 such axioms (1.3%) in nineteen documents—in particular, 300 of these pertain to rule prp-spo1 (non-authoritative subject for subproperty axiom), 404 relate to rules prp-inv1/prp-inv2 (non-authoritative subject/object for inverse-of axioms), and the remaining 300 were spread over eight other rules.

Concluding with Table 5.2, we see that 81.7% of all possible T-ground OWL 2 RL/RDF rules are authoritative *and* supported by our A-linear fragment, and that we *fully* support 51,369 (90.6%) of the 56,710 documents contributing unique terminological axioms.

However, we have yet to consider the importance of these terminological documents. Along these lines, we reuse the ranks for documents computed in § 4.3—again, these ranks are based on a PageRank analysis, denoting the (Eigenvector) centrality of the documents with respect to their linkage on the Web of Data. Thereafter, Table 5.5 presents the sum of ranks for documents featuring each type of axiom: note that the position shown in the left hand column only counts rules requiring identical terminological axioms once (e.g., prp-inv1/prp-inv2), but counts different forms of the same axiom separately (e.g., cls-maxc1/cls-maxc2 which deal with max-cardinalities of zero and one respectively).

Also shown are the positions of the top-ranked document containing such an axiom: note that these positions are relative to the 86,179 documents containing terminology—we provide a legend for the documents with notable positions (higher than 10,000) separately in Table 5.6.[26]

We make the following observations:

1. The top four axioms equate to the core RDFS primitives (or ρDF; see [Muñoz et al., 2009]).[27]

2. Of the top thirteen axiom types, twelve axioms are expressible as a single triple (the exception is `owl:unionOf` in position 8).

3. Of the axioms considered, all twelve RDFS/OWL 1 axioms expressible as a single triple appear in the top thirteen (the exception is again `owl:unionOf` in position 8).

4. The eleven axiom-types that form the RDFS plus profile [Allemang and Hendler, 2008] are in the top thirteen (the remaining two are `owl:disjointWith` in position 6 and `owl:unionOf` in position 8).

5. All eleven axiom types using new OWL 2 constructs are in the bottom twelve—we conjecture that they haven't had time to find proper traction on the Web yet.

6. The total summation of ranks for documents containing *some* axioms which we do not support is 23% of the total summation of document ranks; the highest ranked document which we do not fully support is SKOS (#5) which uses `owl:FunctionalProperty`, `owl:disjointWith` and `owl:TransitiveProperty`.

7. The total summation of ranks for documents containing *some* non-authoritative axioms was 6.7% of the total summation of ranks. The highest ranked non-authoritative axioms were given by FOAF (#7), who publish equivalence relations between `foaf:Agent` and `dcterms:Agent` using `owl:equivalentClass`, and between `foaf:maker` and `dcterms:creator` using `owl:equivalentProperty`: these were

[26] Also note that these do not directly correspond to the rank positions listed in Table 4.5, wherein the DC Elements and RDFS-More documents (#3 and #5 resp.) do not contain any OWL 2 RL/RDF terminology. We were surprised to note that the former document does not contain any terminology: it does contain memberships of `rdf:Property` and RDFS "annotation" properties, but these are not considered as terminology with respect to OWL 2 RL/RDF rules.

[27] Also see `http://web.ing.puc.cl/~jperez/talks/eswc07.pdf`; retr. 2011/01/11

#	Rule ID	Sup.?	Meta-Class/-Property	Sum of Doc. Ranks			Top Ranked Doc.		
				all	auth	¬auth	all	auth	¬auth
1	cax-sco	✓	rdfs:subClassOf	2.95E-01	2.95E-01	3.53E-03	1	1	10
2	prp-rng	✓	rdfs:range	2.94E-01	2.93E-01	3.31E-03	1	1	10
3	prp-dom	✓	rdfs:domain	2.92E-01	2.92E-01	3.85E-03	1	1	10
4	prp-spo1	✓	rdfs:subPropertyOf	8.95E-02	8.92E-02	2.87E-03	4	4	10
5	prp-fp	C	owl:FunctionalProperty	6.27E-02	6.23E-02	2.79E-03	5	5	10
6	cax-dw	⊥	owl:disjointWith	4.85E-02	4.80E-02	3.02E-07	5	5	13,742
7	prp-inv1	✓	owl:inverseOf	4.72E-02	4.68E-02	2.75E-03	5	5	10
	prp-inv2	✓	owl:inverseOf	4.72E-02	4.68E-02	3.25E-03	5	5	10
8	cls-uni	✓	owl:unionOf	3.46E-02	3.45E-02	3.82E-03	5	5	10
9	prp-symp	✓	owl:SymmetricProperty	3.33E-02	2.74E-03	2.80E-03	5	5	10
10	prp-trp	X	owl:TransitiveProperty	3.00E-02	3.00E-02	1.05E-07	5	5	48,058
11	cax-eqc1	✓	owl:equivalentClass	2.11E-02	2.06E-02	2.80E-03	7	7	10
	cax-eqc2	✓	owl:equivalentClass	2.11E-02	6.13E-03	1.52E-02	7	10	7
12	prp-ifp	C	owl:InverseFunctionalProperty	1.88E-02	1.88E-02	3.74E-07	7	7	12,094
13	prp-eqp1	✓	owl:equivalentProperty	1.86E-02	1.82E-02	2.80E-03	7	7	10
	prp-eqp2	✓	owl:equivalentProperty	1.86E-02	3.56E-03	1.48E-02	7	10	7
14	cls-svf1	X	owl:someValuesFrom	1.79E-02	1.75E-02	3.58E-07	6	6	39,002
15	cls-hv1	✓	owl:hasValue	2.86E-04	2.86E-04	–	71	71	–
	cls-hv2	✓	owl:hasValue	2.86E-04	1.61E-04	1.25E-04	71	71	–
16	cls-avf	X	owl:allValuesFrom	2.62E-04	2.62E-04	–	69	69	–
17	cls-maxc2	C	owl:maxCardinality	2.12E-04	2.12E-04	–	71	71	–
18	cls-int1	X	owl:intersectionOf	1.73E-04	1.73E-04	–	91	91	–
	cls-int2	✓	owl:intersectionOf	1.73E-04	1.73E-04	–	91	91	–
19	cls-svf2	✓	owl:someValuesFrom	2.33E-07	2.33E-07	–	10,381	10,381	–
20	cls-com	⊥	owl:complementOf	1.62E-07	1.62E-07	–	29,075	29,075	–
21	cax-adc	⊥	owl:AllDisjointClasses	1.51E-07	1.51E-07	6.17E-08	51,914	51,914	51,914
22	prp-irp	⊥	owl:IrreflexiveProperty	5.94E-08	5.94E-08	5.03E-08	53,653	53,653	63,368
23	prp-asyp	⊥	owl:AsymmetricProperty	5.94E-08	5.94E-08	5.03E-08	53,653	53,653	63,368
24	prp-key	X̸	owl:hasKey	4.99E-08	4.99E-08	–	63,972	63,972	–
25	prp-spo2	X	owl:propertyChainAxiom	4.48E-08	4.48E-08	–	72,343	72,343	–
–	cls-maxc1	⊥	owl:maxCardinality	–	–	–	–	–	–
–	cls-maxqc1	⊥	owl:maxQualifiedCardinality	–	–	–	–	–	–
–	cls-maxqc2	⊥	owl:maxQualifiedCardinality	–	–	–	–	–	–
–	cls-maxqc3	X̸	owl:maxQualifiedCardinality	–	–	–	–	–	–
–	cls-macqc4	C	owl:maxQualifiedCardinality	–	–	–	–	–	–
–	prp-adp	⊥	owl:AllDisjointProperties	–	–	–	–	–	–
–	prp-pdw	⊥	owl:propertyDisjointWith	–	–	–	–	–	–
all rules				3.05E-01	3.05E-01	2.05E-02	1	1	7
rules supported (✓)				3.05E-01	3.05E-01	2.05E-02	1	1	7
rules not supported (X/⊥/C/X̸)				7.01E-02	7.01E-02	2.79E-03	5	5	10

Table 5.5: Summary of ranks of documents in our corpus serving terminological axioms pertaining to OWL 2 RL/RDF rules with non-empty TBody and ABody

added in December 2009,[28] and mandate the translation of the external DC terms into local FOAF terms by rules cax-eqc2 and prp-eqp2 respectively. (Note that translation from FOAF terms to DC terms is still authoritative by rules cax-eqc1 and prp-eqp1.) Further, we note that DC reciprocated these equivalence claims on October 2010 (too late for our crawl in May 2010). Non-authoritative axioms given by the Music Ontology (#10) were due to a misconfiguration of their server, leading to incorrect handling of redirects.[29]

In summary, 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

[28] See `http://xmlns.com/foaf/spec/20091215.html`; retr. 2011/01/11

[29] See `http://groups.google.com/group/pedantic-web/browse_thread/thread/21fa1b2a85d2db44`; retr. 2011/01/11

#	Document	Rank
1	`http://www.w3.org/1999/02/22-rdf-syntax-ns`	1.12E-01
4	`http://dublincore.org/2008/01/14/dcterms.rdf`	3.21E-02
5	`http://www.w3.org/2009/08/skos-reference/skos.rdf`	2.82E-02
6	`http://www.w3.org/2003/g/data-view`	1.42E-02
7	`http://xmlns.com/foaf/spec/`	1.41E-02
10	`http://motools.sourceforge.net/doc/musicontology.rdfs`	2.74E-03
69	`http://www.w3.org/2006/03/wn/wn20/schemas/wnfull.rdfs`	1.65E-04
71	`http://www.w3.org/2006/time`	1.60E-04
91	`http://motools.sourceforge.net/timeline/timeline.rdf`	1.25E-04

Table 5.6: Legend for notable documents (pos.$< 10,000$) whose rank positions are mentioned in Table 5.5

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 see that the top-ranked documents favour OWL 1 axioms which are expressible as a single RDF triple, and that the highest ranked document serving non-authoritative axioms was FOAF (#7).

Authoritative Reasoning

In order to demonstrate the *effects* of (non-)authoritative reasoning wrt. our $\mathcal{O}2\mathcal{R}^{\propto\mathsf{A}}$ rules and corpus, we applied reasoning over the top ten asserted classes and properties. For each class c, we performed reasoning—wrt. the T-ground program and the authoritatively T-ground program—over a single assertion of the form (`x`, `rdf:type`, `c`) where `x` is an arbitrary unique name; for each property p, we performed the same over a single assertion of the form (`x1`, `p`, `x2`).[30] Table 5.7 gives the results (cf. older results in [Hogan et al., 2009b]).[31] Notably, the 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`;[32] thus, in the table we also provide results for the core top-level concepts and `rdf:type`, and provide equivalent counts for inferences not relating to these concepts—still, for these popular terms, non-authoritative inferencing creates 12.74× more inferences than the authoritative equivalent.

We now compare authoritative and non-authoritative inferencing in more depth 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 leads to five authoritative inferences and twenty-six *additional* non-authoritative inferences (all class memberships). Of the latter twenty-six, fourteen are anonymous classes. Table 5.8 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

[30] Subsequently, we only count inferences mentioning an individual name `x*`.

[31] Note that the count of classes and properties is not necessarily unique, where we performed a count of the occurrences of each term in the object of an `rdf:type` triple (class membership) or predicate position (property membership) in our corpus.

[32] 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).

#	Term	n	**a**	n * **a**	$\mathbf{a}^-$	n * $\mathbf{a}^-$	**na**	n * **na**	$\mathbf{na}^-$	n * $\mathbf{na}^-$
Core classes (∼top-level concepts)										
–	`rdfs:Resource`	12,107	0	0	0	0	108	1,307,556	0	0
–	`owl:Thing`	679,520	1	679,520	0	0	109	74,067,680	0	0
Core property										
–	`rdf:type`	206,799,100	1	206,799,100	0	0	109	22,541,101,900	0	0
Top ten asserted classes										
1	`foaf:Person`	163,699,161	6	982,194,966	5	818,495,805	140	22,917,882,540	31	5,074,673,991
2	`foaf:Agent`	8,165,989	2	16,331,978	1	8,165,989	123	1,004,416,647	14	114,323,846
3	`skos:Concept`	4,402,201	5	22,011,005	3	13,206,603	115	506,253,115	6	26,413,206
4	`mo:MusicArtist`	4,050,837	1	4,050,837	0	0	132	534,710,484	23	93,169,251
5	`foaf:PersonalProfileDocument`	2,029,533	2	4,059,066	1	2,029,533	114	231,366,762	5	10,147,665
6	`foaf:OnlineAccount`	1,985,390	2	3,970,780	1	1,985,390	110	218,392,900	1	1,985,390
7	`foaf:Image`	1,951,773	1	1,951,773	0	0	110	214,695,030	1	1,951,773
8	`opiumfield:Neighbour`	1,920,992	1	1,920,992	0	0	109	209,388,128	0	0
9	`geonames:Feature`	983,800	2	1,967,600	1	938,800	111	109,201,800	2	1,967,600
10	`foaf:Document`	745,393	1	745,393	0	0	113	84,229,409	4	2,981,572
Top ten asserted properties (after `rdf:type`)										
1	`rdfs:seeAlso`	199,957,728	0	0	0	0	218	43,590,784,704	0	0
2	`foaf:knows`	168,512,114	14	2,359,169,596	12	2,022,145,368	285	48,025,952,490	67	11,290,311,638
3	`foaf:nick`	163,318,560	0	0	0	0	0	0	0	0
4	`bio2rdf:linkedToFrom`	31,100,922	0	0	0	0	0	0	0	0
5	`lld:pubmed`	18,776,328	0	0	0	0	0	0	0	0
6	`rdfs:label`	14,736,014	0	0	0	0	221	3,256,659,094	3	44,208,042
7	`owl:sameAs`	11,928,308	5	59,641,540	1	11,928,308	221	2,636,156,068	3	35,784,924
8	`foaf:name`	10,192,187	5	50,960,935	2	20,384,374	256	2,609,199,872	38	387,303,106
9	`foaf:weblog`	10,061,003	8	80,488,024	5	50,305,015	310	3,118,910,930	92	925,612,276
10	`foaf:homepage`	9,522,912	8	76,183,296	5	47,614,560	425	4,047,237,600	207	1,971,242,784
total		**1,035,531,872**	**65**	**3,873,126,401**	**37**	**2,997,244,745**	**3,439**	**155,931,914,709**	**497**	**19,982,077,064**

Table 5.7: Summary of authoritative inferences vs. non-authoritative inferences for core properties, classes, and top-ten most frequently asserted classes and properties: given are the number of asserted memberships of the term n, the number of unique inferences (which mention an "individual name") possible for an arbitrary membership assertion of that term wrt. the authoritative T-ground program (**a**), the product of the number of assertions for the term and authoritative inferences possible for a single assertion (n * **a**), respectively, the same statistics excluding inferences involving the top-level concepts `rdfs:Resource` and `owl:Thing` ($\mathbf{a}^-$ | n * $\mathbf{a}^-$), statistics for non-authoritative inferencing (**na** | n * **na**) and also non-authoritative inferences minus inferences through a top-level concept ($\mathbf{na}^-$ | n * $\mathbf{na}^-$)

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

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 § 5.1.4)—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 5.9 gives the results for reasoning on one machine for each approach outlined in § 5.3.4. 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 5.1, 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[33]
Pi	8.9	22.1
Pim	8.9	17.7
Pims	8.9	19.5

Table 5.9: Performance for reasoning over 1.1 billion statements on one machine for all approaches

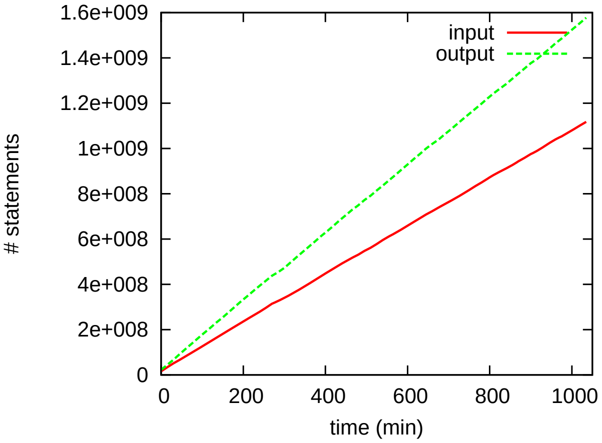


Figure 5.1: Detailed throughput performance for application of assertional program using the fastest approach: Pim

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 5.10: Distributed reasoning in *minutes* using Pim for 1, 2, 4 & 8 slave machines

Distributed Reasoning

We also apply reasoning over 1, 2, 4 and 8 slave machines using the distribution strategy outlined in § 5.4.3; Table 5.10 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 1.118 billion Linked Data triples in 3.35 h using nine machines, deriving 1.58 billion inferred triples, of which 962 million are novel and unique.

5.5 Related Work

Herein, we discuss related works specifically in the field of scalable and distributed reasoning (§ 5.5.1) as well as works in the area of robust Web reasoning (§ 5.5.2).

5.5.1 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 [Bishop et al., 2011], 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.[34] 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.[35] BigOWLIM features distribution, but only as a replication strategy for fault-tolerance and supporting higher query load.

Following our initial work on SAOR—which had discussed the benefits of a separation of terminological data, but which had not demonstrated distribution [Hogan et al., 2009b]—a number of scalable and distributed reasoners adopted a similar approach.

Weaver and Hendler [2009] 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. [2009] use MapReduce [Dean and Ghemawat, 2004] for distributed RDFS materialisation over 850 million 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 Weaver and Hendler [2009]—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 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 [Urbani et al., 2010] 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 [Hogan et al., 2009b] (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.*"

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

[35] `http://www.readwriteweb.com/archives/bbc_world_cup_website_semantic_technology.php`; retr. 2011/01/22

—[Urbani et al., 2010]

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, Kolovski et al. [2010] 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, Kolovski et al. [2010] 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.

5.5.2 Web Reasoning

As previously mentioned, Urbani et al. [2009] discuss reasoning over 850m Linked Data triples—however, they only do so over RDFS and do not consider any issues relating to provenance.

Kiryakov et al. [2009] apply reasoning over 0.9 billion Linked Data triples using the aforementioned BigOWLIM reasoner; however, their "LDSR" dataset is comprised of a small number of manually selected datasets, as opposed to an arbitrary corpus—they do not consider any general notions of provenance or Web tolerance. (Again, Urbani et al. [2010] 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., [Ghilardi et al., 2006; Lutz et al., 2007; Jiménez-Ruiz et al., 2008]). 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. However, defining `foaf:name` to be a sub-property of `foaf:givenName` outside of the FOAF vocabulary would be disallowed by both authoritative reasoning and model conservative extensions since new inferences using FOAF terms could be created. Summarising, we can state that (on an abstract level) all cases of non-conservative extension are cases of non-authoritative definitions, but not vice versa: some non-authoritative definitions may be conservative extensions.[36] Finally, we note that works on conservative extension focus moreso on scenarios involving few ontologies within a "curated" environment, and do not consider the Web use-case, or, for example, automatic analyses based on Linked Data publishing principles.

In a similar approach to our authoritative analysis, Cheng and Qu [2008] introduced restrictions for accepting sub-class and equivalent-class axioms from third-party sources; they follow similar arguments to that made in this thesis. However, their notion of what we call authoritativeness is based on hostnames and does not consider redirects; we argue that both simplifications are not compatible with the common use of PURL services[37]: (i) all documents using the same service (and having the same namespace hostname) would

[36] Informally, we note that non-conservative extension can be considered "harmful" hijacking which contravenes **robustness**, whereas the remainder of ontology hijacking cases can be considered "inflationary" and morseo contravening **terseness**.

[37] `http://purl.org/`; retr. 2011/01/14; see Table A.2 for 10 namespaces in this domain, including `dc:` and `dcterms:`.

be 'authoritative' for each other, (ii) the document cannot be served directly by the namespace location, but only through a redirect. Indeed, further work presented by Cheng et al. [2008b] better refined the notion of an *authoritative description* to one based on redirects—and one 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 Delbru et al. [2008] and employed by Sindice [Oren et al., 2008]. 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. In contrast to our approach where we construct one authoritative terminological model for all Web data, their approach uses a bespoke trusted model for each document; thus, they would infer statements within the local context which we would consider to be non-authoritative, but our model is more flexible for performing inference over the merge of documents.[38] As such, they also consider a separation of terminological and assertional data; in this case ontology documents and data documents. 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.

5.6 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 at the start of the chapter, 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 § 5.4.4, 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. Current vocabularies popular on the Web—such as Dublin Core, FOAF and SIOC—are very much open to community feedback and suggestions, and commonly map between each other as appropriate. However, this may not be so true of more niche or fringe vocabularies; one could

[38] Although it should be noted that without considering rules with assertional joins, our ability to make inferences across documents is somewhat restricted; however, we will be looking at the application of such rules for supporting equality in Chapter 7.

imagine the scenario whereby a vocabulary achieves some adoption, but then falls out of maintenance and the community provides mappings in a separate location. Thus, in future work, we believe it would be worthwhile to investigate "trusted" third-party mappings in the wild, perhaps based on links-analysis or observed adoption.[39]

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 will discuss such an approach for equality reasoning in Chapter 7. 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. 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 [Urbani et al., 2010]); 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. We will examine this issue further in the next chapter.

[39] This may depend on more philosophical considerations as to whether showing special favour to established, well-linked vocabularies is appropriate. Our authoritative reasoning is deliberately quite *democratic*, and does not allow popular vocabularies to redefine smaller vocabularies; each vocabulary has its own guaranteed "rights and privileges".

Chapter 6

Annotated Reasoning[*]

"Logic is the art of going wrong with confidence."

—Joseph Wood Krutch

In the previous chapter, we looked at performing reasoning with respect to a scalable subset of OWL 2 RL/RDF rules over a corpus of arbitrarily sourced Linked Data. Although we demonstrated the approach to be feasible with respect to our evaluation corpus, we informally noted that reasoning may still introduce unintended consequences and accentuate various types of noise.

In this chapter, we want to move away from crisp, binary truth values—where something is either true or false—to truth values which better capture the unreliable nature of inferencing over Web data, and offer varying degrees of the *strength* of a particular derivation. Thus, we look at incorporating more fine-grained information about the underlying corpus within the reasoning framework; to do this, we use annotations and other concepts from the field of Annotated Logic Programs [Kifer and Subrahmanian, 1992].

We thus derive a formal logical framework for annotated reasoning in our setting; within this framework, we encode the notion of authority from the previous chapter, as well as a simple annotation for blacklisting triples and an annotation which includes a rank value for each triple computed from links-based analysis of the sources in the corpus. We then demonstrate a use-case of the latter form of annotation, using OWL 2 RL/RDF inconsistency detection rules to pinpoint (a subset of) noise in the materialisation-extended corpus, and to subsequently perform a repair thereof.

Thus, this chapter is organised as follows:

- we begin by discussing some concepts relating to General Annotated Programs which inspire our formalisms and lend useful results (§ 6.1);
- we introduce the three annotation values we currently consider for our Linked Data use-case (§ 6.2);
- we formalise our annotated program framework, demonstrating some computational properties for various reasoning tasks (§ 6.3);
- we discuss the distributed implementation and evaluation of our methods for ranking, reasoning and repairing Linked Data corpora (§ 6.4);
- we present related works in the field of annotated reasoning, knowledgebase repair, and reasoning in the presence of inconsistency (§ 6.5);

[*] Parts of this chapter have been preliminarily accepted for publication as [Bonatti et al., 2011].

- we conclude the chapter with discussion, critical appraisal, and future directions (§ 6.6).

Note that in the following, our formalisms allow for extension towards other (possibly multi-dimensional) domains of annotation one might consider useful for reasoning over Web data.

6.1 Generalised Annotated Programs

Herein, we introduce key concepts from the works of Kifer and Subrahmanian [1992] on *Generalised Annotated Programs*, which form the basis of our (more specialised) framework.

In Generalised Annotated Programs, annotations are used to represent an extent to which something is true. This set of truth values can take the form of any arbitrary upper semilattice $\mathcal{T}$: a partially ordered set over which any subset of elements has a defined *greatest lower bound* (or *glb*, *infimum*—the greatest element in $\mathcal{T}$ which is known to be less than or equal to all elements of the subset). Such a semilattice can represent truth-values from arbitrary domains, such as time intervals, geo-spatial regions, provenance, probabilistic or fuzzy values, etc. Atoms with truth-values from $\mathcal{T}$ (or a variable truth-value ranging over $\mathcal{T}$, or a function over such truth values) are called *annotated atoms*:

Definition 6.1 (Generalised annotated atoms) *We denote* annotated atoms *by $A{:}\mu$ where μ is either (i) a simple annotation: an element of the semilattice $\mathcal{T}$, or a variable ranging over $\mathcal{T}$; or (ii) a complex annotation: a function $f(\mu_1, \ldots, \mu_n)$ over a tuple of simple annotations.*

Thereafter, Generalised Annotated Programs allow for doing reasoning over data annotated with truth values of this form using *annotated rules*:

Definition 6.2 (Generalised annotated rules) Annotated rules *are expressions of the form:*

$$H{:}\rho \leftarrow B_1{:}\mu_1, \ldots, B_n{:}\mu_n \,,$$

where H, B_1, B_n are atoms (as per § 3.4), where all $\mu_i (1 \leq i \leq n)$ are simple annotations and where ρ is a complex annotation of the form $f(\mu_1, \ldots, \mu_n)$.

Thus, annotated rules are rules as before, but which additionally apply some function over the set of annotations in the instance of the body to produce the annotation of the consequence. Note that complex annotations are only allowed in the head of the rule, where variables appearing in this annotation must also appear as annotations of the body atoms [Kifer and Subrahmanian, 1992].

Other annotated programming concepts—such as facts, programs, etc.—follow naturally from their classical (non-annotated) version, where facts are associated with constant annotations and programs are sets of annotated rules (and facts).

Moving forward, *restricted interpretations* map each ground atom to a member of $\mathcal{T}$.

Definition 6.3 (Restricted interpretations) *A* restricted interpretation *I satisfies $A{:}\mu$ (in symbols, $I \models A{:}\mu$) iff $I(A) \geq_{\mathcal{T}} \mu$, where $\geq_{\mathcal{T}}$ is $\mathcal{T}$'s ordering.*

From this notion of a restricted interpretation follows the restricted immediate consequence operator of a general annotated program P:

Definition 6.4 (Restricted immediate consequences) *The* restricted immediate consequence operator *is given as follows:*

$$\mathfrak{R}_P(I)(H) = \mathsf{lub}\big\{\rho \mid (H{:}\rho \leftarrow B_1{:}\mu_1, \ldots, B_n{:}\mu_n)\sigma \in \mathsf{Ground}(P), I \models (B_i{:}\mu_i)\sigma \textit{ for } (1 \leq i \leq n)\big\} \,,$$

where σ is a substitution for annotation variables, $\mathsf{Ground}(P)$ *is a shortcut for classical rule instantiation (as per § 3.4, with a slight abuse of notation to ignore annotations), and the function* lub *returns the least upper bound (or supremum—the least element in* $\mathcal{T}$ *which is known to be greater than or equal to all elements of the set) of a set of ground annotations in* $\mathcal{T}$*.*

Note that the ρ function is always considered evaluable, and so when all μ_i are substituted for constant annotations (necessary for $I \models (B_i{:}\mu_i)$ to hold), ρ will evaluate to a constant annotation.

Kifer and Subrahmanian [1992] demonstrated various desirable and (potentially) undesirable properties of $\mathfrak{R}_P$; for example, they discussed how $\mathfrak{R}_P$ is monotonic, but not always *continuous*: loosely, a continuous function is one where there are no "impulses" in the output caused by a small change in the input, where in particular, $\mathfrak{R}_P$ may not be continuous if a rule body contains a mix of annotation variables and annotation constants (we will see an example later in § 6.3.2), and where given discontinuity, $\mathsf{lfp}(\mathfrak{R}_P) = \mathfrak{R}_P \uparrow \omega$ does not always hold.

We will leverage these formalisms and results as the basis of our more specialised annotated programs—we will be looking at them in more detail later, particularly in § 6.3. First, we introduce the annotation values we wish to track for our Linked Data use-case.

6.2 Use-case Annotations

Moving forward, in this section we discuss the three annotation values we have chosen to represent a combined truth value within our specialised annotated programs for reasoning over Linked Data: *blacklisting*, *authority*, and *triple rank*. These values are combined and processed during the annotated reasoning procedure to produce annotations for inferred triples.

6.2.1 Blacklisting

Despite our efforts to create algorithms which automatically detect and mitigate noise in the input corpus, it may often be desirable to *blacklist* input data or derived data based on some (possibly heuristic) criteria: for example, data from a certain domain may be considered likely to be spam, or certain triple patterns may constitute common publishing errors which hinder the reasoning process. We currently do not require the blacklisting function, and thus consider all triples to be *not blacklisted* per default. However, such an annotation has obvious uses for bypassing noise which cannot otherwise be automatically detected, or which can occur during the reasoning process.

One such example we had in mind was for blacklisting void values for inverse-functional properties, whereby publishers give empty literal values for properties such as `foaf:mbox_sha1sum`, or generic URI values such as `http://facebook.com/` for `foaf:homepage`—however, in our formal reasoning framework, we currently do not include the specific OWL 2 RL/RDF rule (prp-ifp; Table B.8) which would infer the incorrect `owl:sameAs` relations caused by such noise since it contains more than one assertional atom, and thus falls outside of our scalable subset. Instead, rules relating to equality are supported using bespoke optimisations discussed separately in Chapter 7; therein, the most common void values for inverse-functional properties is listed in Table 7.5.

In summary, the blacklisting annotation essentially serves as a pragmatic *last resort* for annotating data considered to be noise: data which should be circumvented during inferencing.

6.2.2 Authoritative Analysis

As discussed in § 5.4.2, our reasoning framework includes consideration of the provenance of terminological data, conservatively excluding certain third-party (unvetted) contributions. In this chapter, we demonstrate how such values can be included within the formalisms of our annotation framework.

6.2.3 Triple Ranks

The primary motivation for investigating annotations is to incorporate ranks of individual triples into the reasoning process. Later in this chapter, we will provide a use-case for these ranks relating to the repair of inconsistencies, but one can also imagine scenarios whereby consumers can leverage the ranks of input and inferred triples for the purposes or prioritising the display of information in a user interface, etc.

First, we need to annotate the input triples. To do so, we reuse the ranks of sources calculated in § 4.3: we calculate the ranks for individual triples as the summation of the ranks of sources in which they appear, based on the intuition that triples appearing in highly ranked sources should benefit from that rank, and that each additional source stating a triple should increase the rank of the triple.[1] Thus, the process for calculating the rank of a triple t is simply as follows:

$$\mathsf{trank}(t) = \sum_{s_t \in \{s \in \mathsf{S} | t \in \mathsf{get}(s)\}} \mathsf{rank}(s_t)\,.$$

In particular, we note that core information about resources is often repeated across data sources, where, for example, in profiles using the FOAF vocabulary, publishers will often assert that their acquaintances are members of the class `foaf:Person` and provide their name as a value for the property `foaf:name`; thus, our ranking scheme positively rewards triples for being re-enforced across different documents. Relatedly, from the statistics of our corpus presented in § 4.2.2, we note that of our 1.106 billion *unique* quadruples, 947 million are unique triples, implying that 14.4% of our corpus is composed of triples which are repeated across documents.

Note that we will discuss the (straightforward) implementation for annotating the input corpus with these ranking annotations in § 6.4.1.

6.3 Formal Annotation Framework

In this section, we look at incorporating the above three dimensions of trust and provenance—blacklisting, authority and triple-rank, which we will herein refer to as *annotation properties*—into a specialised annotated logic programming framework which tracks this information during reasoning, and determines the annotations of inferences based on the annotations of the rule and the relevant instances, where the resultant values of the annotation properties can be viewed as denoting the strength of a derivation (or as a truth value).

6.3.1 Annotation Domains

The annotation properties are abstracted by an arbitrary finite set of domains $D_1, \ldots, D_z$:

Definition 6.5 (Annotated domain) *An* annotation domain *is a cartesian product* $\mathbf{D} = \times_{i=1}^{z} D_i$ *where each* D_i *is totally ordered by a relation* $\leq_i$ *such that each* D_i *has a* $\leq_i$*-maximal element* $\top_i$*. Define a partial order* $\leq$ *on* $\mathbf{D}$ *as the direct product of the orderings* $\leq_i$*, that is* $\langle d_1, \ldots, d_z \rangle \leq \langle d'_1, \ldots, d'_z \rangle$ *iff for all* $1 \leq i \leq z$*,* $d_i \leq_i d'_i$.[2] *When* $\langle d_1, \ldots, d_z \rangle < \langle d'_1, \ldots, d'_z \rangle$ *we say that* $\langle d'_1, \ldots, d'_z \rangle$ dominates $\langle d_1, \ldots, d_z \rangle$.[3]

[1] Note that one could imagine a spamming scheme where a large number of spurious low-ranked documents repeatedly make the same assertions to create a set of highly-ranked triples. In future, we may revise this algorithm to take into account some limiting function derived from PLD-level analysis.

[2] We favour angle brackets to specifically denote a tuple of annotation values.

[3] Note that we thus do not assume a lexicographical order.

We denote with $\mathsf{lub}(\mathbf{D}')$ *and* $\mathsf{glb}(\mathbf{D}')$ *respectively the least upper bound and the greatest lower bound of a subset* $\mathbf{D}' \subseteq \mathbf{D}$.

For the use-case annotation domain based on blacklisting, authoritativeness, and ranking, $z = 3$ and $D_1 = \{\mathbf{b}, \mathbf{nb}\}$ (**b**=blacklisted, **nb**=non-blacklisted), $D_2 = \{\mathbf{na}, \mathbf{a}\}$ (**a**=authoritative, **na**=non-authoritative), $D_3 = \mathbb{R}$. Moreover, $\mathbf{b} \leq_{\mathbf{1}} \mathbf{nb}$, $\mathbf{na} \leq_{\mathbf{2}} \mathbf{a}$, and $x \leq_{\mathbf{3}} y$ iff $x \leq y$.

6.3.2 (Specialised) Annotated Programs

Following our definition of the domain of annotations, (specialised) annotated programs are defined as follows:

Definition 6.6 (Annotated programs) *An* annotated program *P is a finite set of annotated rules*

$$H \leftarrow B_1, \ldots, B_m : \mathbf{d} \qquad (m \geq 0)$$

where $H, B_1, \ldots, B_m$ are logical atoms and $\mathbf{d} \in \mathbf{D}$. When $m = 0$, a rule is called a fact and denoted by H:$\mathbf{d}$ (omitting the arrow).

Note that again, any (simple) predicate can be considered for the atoms, but in practice we will only be using an implicit ternary (triple) predicate ($\mathtt{s}$, $\mathtt{p}$, $\mathtt{o}$). As opposed to the formalisms for Generlised Annotated Programs, our annotated programs associate each rules (or fact) with a constant annotation.

Now we can define the models of our programs. The semantics of a fact F is a set of annotations, covering the possible ways of deriving F. Roughly speaking, the annotations of F include the "minimum" of the annotations which hold for the facts and rule(s) from which F can be inferred.

Definition 6.7 (Annotation interpretations) *Let $\mathcal{B}_P$ be the Herbrand base of a program P (the set of all possible facts from the constants in P). An* annotation interpretation *is a mapping $I : \mathcal{B}_P \to 2^{\mathbf{D}}$ that associates each fact $F \in \mathcal{B}_P$ with a set of possible annotations.*

Given a ground rule R of the form $H \leftarrow B_1, \ldots, B_m : \mathbf{d}$ an interpretation I satisfies R if for all $\mathbf{d}_i \in I(B_i)$ $(1 \leq i \leq m)$, $\mathsf{glb}(\{\mathbf{d}_1, \ldots, \mathbf{d}_m, \mathbf{d}\}) \in I(H)$.

More generally, I satisfies a (possibly non-ground) rule R (in symbols, $I \models R$) iff I satisfies all of the ground rules in $\mathsf{Ground}(R)$. Accordingly, I is a model of a program P ($I \models P$) iff for all $R \in P$, $I \models R$. Finally, we say that the fact F:$\mathbf{d}$ is a logical consequence of P (written $P \models F$:$\mathbf{d}$) iff for all interpretations I, $I \models P$ implies $I \models F$:$\mathbf{d}$.

Following the same principles as for our notion of a classical program, we can define an immediate consequence operator $\mathfrak{A}_P$ for annotated programs (of the form described in Definition 6.6) as follows:

Definition 6.8 (Annotation immediate consequences) *The* annotation immediate consequence operator *is a mapping over annotation interpretations such that for all facts $F \in \mathcal{B}_P$:*

$$\mathfrak{A}_P(I)(F) = \bigcup_{F \leftarrow B_1, \ldots, B_m : \mathbf{d} \in \mathsf{Ground}(P)} \big\{ \mathsf{glb}(\{\mathbf{d}_1, \ldots, \mathbf{d}_m, \mathbf{d}\}) \mid \forall_{1 \leq i \leq m} \big(\mathbf{d}_i \in I(B_i)\big) \big\}$$

6.3.3 Least Fixpoint and Decidability

We now demonstrate that the semantics of annotated programs have the same desirable properties as those for our classical program: every given annotated program P has one minimal model which contains exactly the logical consequences of P, and which can be characterised as the least fixed point of the monotonic immediate consequence operator $\mathfrak{A}_P$.

To see this, we first need to define a suitable ordering over interpretations:

$$I \preceq I' \Leftrightarrow \forall F \in \mathcal{B}_P \big(I(F) \subseteq I'(F)\big)$$

The partial order $\preceq$ induces a complete lattice in the set of all interpretations. Given a set of interpretations $\mathcal{I}$ the least upper bound $\sqcup\mathcal{I}$ and the greatest lower bound $\sqcap\mathcal{I}$ satisfy $\sqcup\mathcal{I}(F) = \bigcup_{I\in\mathcal{I}} I(F)$ and $\sqcap\mathcal{I}(F) = \bigcap_{I\in\mathcal{I}} I(F)$, for all $F \in \mathcal{B}_P$.[4] The bottom interpretation Δ maps each $F \in \mathcal{B}_P$ to $\emptyset$.

Theorem 6.1 *For all programs P and interpretations I:*

1. *I is a model of P iff $\mathfrak{A}_P(I) \preceq I$;*
2. *$\mathfrak{A}_P$ is monotone, i.e. $I \preceq I'$ implies $\mathfrak{A}_P(I) \preceq \mathfrak{A}_P(I')$.*

Proof: Our framework can be regarded as a special case of General Annotated Programs [Kifer and Subrahmanian, 1992] (introduced in § 6.1). In that framework, our rules can be reformulated as

$$H{:}\rho \leftarrow B_1{:}\mu_1, \ldots, B_n{:}\mu_n$$

where each μ_i is a variable ranging over $2^{\mathbf{D}}$ and

$$\rho := \big\{\mathsf{glb}(\{\mathbf{d}_1, \ldots, \mathbf{d}_m, \mathbf{d}\}) \mid \forall_{1\leq i\leq n}(\mathbf{d}_i \in \mu_i)\big\} \tag{6.1}$$

The upper semilattice of truth values $\mathcal{T}$ [Kifer and Subrahmanian, 1992, § 2] can be set, in our case, to the complete lattice $\langle 2^{\mathbf{D}}, \subseteq, \cup, \cap\rangle$. Then our semantics corresponds to the *restricted* semantics defined in [Kifer and Subrahmanian, 1992] and our operator $\mathfrak{A}_P$ corresponds to the operator $\mathfrak{R}_P$ which has been proven in [Kifer and Subrahmanian, 1992] to satisfy the two statements. □

Corollary 6.2 *For all programs P:*

1. *P has a minimal model that equals the least fixed point of $\mathfrak{A}_P$, $\mathsf{lfp}(\mathfrak{A}_P)$;*
2. *for all $F \in \mathcal{B}_P$, $\mathbf{d} \in \mathsf{lfp}(\mathfrak{A}_P)(F)$ iff $P \models F{:}\mathbf{d}$.*

Another standard consequence of Theorem 6.1 is that $\mathsf{lfp}(\mathfrak{A}_P)$ can be calculated in a bottom-up fashion, starting from the empty interpretation Δ and applying iteratively $\mathfrak{A}_P$. Define the iterations of $\mathfrak{A}_P$ in the usual way: $\mathfrak{A}_P \uparrow 0 = \Delta$; for all ordinals α, $\mathfrak{A}_P \uparrow (\alpha+1) = \mathfrak{A}_P(\mathfrak{A}_P \uparrow \alpha)$; if α is a limit ordinal, let $\mathfrak{A}_P \uparrow \alpha = \sqcup_{\beta<\alpha}\mathfrak{A}_P \uparrow \beta$. Now, it follows from Theorem 6.1 that there exists an α such that $\mathsf{lfp}(\mathfrak{A}_P) = \mathfrak{A}_P \uparrow \alpha$.

To ensure that the logical consequences of P can be effectively computed, it should also be proven that $\alpha \leq \omega$—in other words that $\mathfrak{A}_P \uparrow \omega = \mathsf{lfp}(\mathfrak{A}_P)$—which is usually done by showing that $\mathfrak{A}_P$ is continuous (§ 6.1). Before we continue, we paraphrase [Kifer and Subrahmanian, 1992, Ex. 3] in order to demonstrate a discontinuous program for which $\mathfrak{R}_P \uparrow \omega = \mathsf{lfp}(\mathfrak{R}_P)$ does *not* hold with respect to *their* restricted immediate consequence operator $\mathfrak{R}_P$:

Example 6.1 *Consider a simple general annotated program P with truth values $\mathcal{T}$ from the set $\{r \in \mathbb{R} \mid 0 \leq r \leq 1\}$ and three rules as follows:*

$$\begin{aligned} A{:}0 &\leftarrow \\ A{:}\tfrac{1+\alpha}{2} &\leftarrow A{:}\alpha \\ B{:}1 &\leftarrow A{:}1 \end{aligned}$$

[4] We favour $\sqcup$ over lub to denote the least upper bound of a *set* of interpretations, where it corresponds with the set-union operator; we favour $\sqcap$ over glb for greatest upper bound analogously.

By the restricted semantics of general annotated programs, $A{:}1 \in \mathfrak{R}_P \uparrow \omega$. However, since the third rule is discontinuous, $B{:}1 \notin \mathfrak{R}_P \uparrow \omega$ and so we see that $\mathfrak{R}_P \uparrow \omega \neq \mathsf{lfp}(\mathfrak{R}_P)$; note that $B{:}1 \in \mathfrak{R}_P \uparrow (\omega + 1)$. $\Diamond$

Thus, even if a general annotated program is Datalog (i.e., it has no function symbols), $\mathfrak{R}_P$ may be discontinuous if a mix of constant and variable annotations are used (as in the example) [Kifer and Subrahmanian, 1992]. In order to prove that *our* $\mathfrak{A}_P$ is continuous, we have to first demonstrate specific properties of the glb function given for ρ in (6.1).

Lemma 6.3 *Let $\mathbf{D}$ be a z-dimensional annotation domain, P a program and F a fact. The number of possible annotations $\mathbf{d}$ such that $P \models F{:}\mathbf{d}$ is bounded by $|P|^z$.*

Proof: Let D_i^P, for $1 \leq i \leq z$, be the set of all values occurring as the i-th component in some annotation in P and $\mathbf{D}^P = \times_{i=1}^{z} D_i^P$. Clearly, for all $i = 1, \ldots, z$, $|D_i^P| \leq |P|$, therefore the cardinality of $\mathbf{D}^P$ is at most $|P|^z$. We are only left to show that the annotations occurring in $\mathfrak{A}_P \uparrow \alpha$ are all members of $\mathbf{D}^P$. Note that if $\{\mathbf{d}_1, \ldots, \mathbf{d}_m, \mathbf{d}\} \subseteq \mathbf{D}^P$, then also $\mathsf{glb}\{\mathbf{d}_1, \ldots, \mathbf{d}_m, \mathbf{d}\} \in \mathbf{D}^P$. Then, by straightforward induction on α, it follows that for all α, if $F{:}\mathbf{d} \in \mathfrak{A}_P \uparrow \alpha$, then $\mathbf{d} \in \mathbf{D}^P$.

In other words, since the glb function cannot introduce new elements from the component sets of the annotation domain, the application of $\mathfrak{A}_P$ can only create labels from the set of tuples which are combinations of existing domain elements in P; thus the set of all labels is bounded by $|P|^z$. □

Next, in order to demonstrate that $\mathfrak{A}_P$ is continuous we must introduce the notion of a *chain of interpretations*: a sequence $\{I_\beta\}_{\beta \leq \alpha}$ such that for all $\beta < \gamma$, $I_\beta \preceq I_\gamma$. Now, $\mathfrak{A}_P$ is continuous if applying $\mathfrak{A}_P$ to the union of the interpretations in a chain is equivalent to the union of applying $\mathfrak{A}_P$ individually to each interpretation in the chain. Formally:

Theorem 6.4 *For all programs P, $\mathfrak{A}_P$ is continuous: that is, for all chains $\mathcal{I} := \{I_\beta\}_{\beta \leq \alpha}$, it holds that $\mathfrak{A}_P(\sqcup \mathcal{I}) = \sqcup\{\mathfrak{A}_P(I) \mid I \in \mathcal{I}\}$.*

Proof: The $\supseteq$ inclusion is trivial since $\mathfrak{A}_P$ is monotone. For the $\subseteq$ inclusion, assume that $\mathbf{d} \in \mathfrak{A}_P(\sqcup \mathcal{I})(H)$. By definition, there exists a rule $H \leftarrow B_1, \ldots, B_n : \mathbf{d}'$ in $\mathsf{Ground}(P)$ and some $\mathbf{d}_1, \ldots, \mathbf{d}_n$ such that $\mathbf{d} = \mathsf{glb}(\mathbf{d}', \mathbf{d}_1, \ldots, \mathbf{d}_n)$ and for all $1 \leq j \leq n$, $\mathbf{d}_j \in \sqcup \mathcal{I}(H)$. Therefore, for all $1 \leq j \leq n$ there exists a $\beta_j \leq \alpha$ such that $\mathbf{d}_j \in I_{\beta_j}$. Let β be the maximum value of $\mathbf{d}_1, \ldots, \mathbf{d}_n$; since $\mathcal{I}$ is a chain, $\mathbf{d}_j \in I_\beta(H)$, for all $1 \leq j \leq n$. Therefore, $\mathbf{d}$ is in $\mathfrak{A}_P(I_\beta)(H)$ and hence in $\sqcup\{\mathfrak{A}_P(I) \mid I \in \mathcal{I}\}(H)$. □

Corollary 6.5 *The interpretation $\mathfrak{A}_P \uparrow \omega$ is the least fixed point of $\mathfrak{A}_P$, $\mathsf{lfp}(\mathfrak{A}_P)$, and hence it is the minimal model of P.*

The logical consequences of our programs satisfy another important property that does not hold for general annotated programs, even if $\mathfrak{R}_P$ is continuous.

Example 6.2 *Consider Example 6.1, but drop the last (discontinuous) rule:*

$$A{:}0 \leftarrow$$

$$A{:}\tfrac{1+\alpha}{2} \leftarrow A{:}\alpha$$

This program is continuous such that, with respect to the restricted semantics of general annotated programs, $\mathfrak{R}_P \uparrow \omega = \mathsf{lfp}(\mathfrak{R}_P)$. However, although $A{:}1 \in \mathfrak{R}_P \uparrow \omega$, it is not finitary because for all $i < \omega$, $A{:}1 \notin \mathfrak{R}_P \uparrow i$.$\Diamond$

Thus, we now need to explicitly demonstrate that the fixpoint of *our* $\mathfrak{A}_P$ is finitary.

Lemma 6.6 *A fact F:$\mathbf{d}$ is a ground logical consequence of P iff for some (finite) $i < \omega$, $\mathbf{d} \in \mathfrak{A}_P \uparrow i(F)$.*

Proof: Due to Corollaries 6.2 and 6.5, $P \models F$:$\mathbf{d}$ iff $\mathbf{d} \in \mathfrak{A}_P \uparrow \omega(F)$. By definition, $\mathfrak{A}_P \uparrow \omega(F) = \bigcup_{i<\omega} \mathfrak{A}_P \uparrow i(F)$, therefore $P \models F$:$\mathbf{d}$ iff for some finite $i < \omega$, $\mathbf{d} \in \mathfrak{A}_P \uparrow i(F)$. □

In the jargon of classical logic, this means that our framework is *finitary*, and that the logical consequences of P are semidecidable. Moreover, if P is Datalog, then the least fixed point of $\mathfrak{A}_P$ is reached after a finite number of iterations:

Lemma 6.7 *If P is Datalog, then there exists $i < \omega$ such that $\mathsf{lfp}(\mathfrak{A}_P) = \mathfrak{A}_P \uparrow i$.*

Proof: Due to Corollary 6.2 and Lemma 6.3, for each $F \in \mathcal{B}_P$, the set of annotations in $\mathsf{lfp}(\mathfrak{A}_P)(F)$ is finite. Moreover, when P is Datalog, the Herbrand base $\mathcal{B}_P$ is finite as well. Thereafter, since $\mathfrak{A}_P$ is monotone, the lemma holds. □

Finally, (and as for our classical program) it follows that the least model of P (again denoted $\mathsf{lm}(P)$) can be finitely represented, and that the logical consequences of P are decidable.

6.3.4 Seeding Annotations

We now briefly discuss a generic formalisation for deriving the initial set of annotations for the base program—that is, the initial A-linear OWL 2 RL/RDF program $\mathcal{O}2\mathcal{R}^{\propto\mathsf{A}}$ (§ 5.4.1; herein, we may refer to these as *meta-rules*) and the input corpus of quadruples.

Recalling our specific annotations of blacklisting, triple ranks, and authority, we can identify three categories of annotation according to the information they require to *seed* from the input:

1. Some—like certain forms of blacklisting—depend on structural properties of input facts (e.g., inverse-functional values set to empty strings or triples with URIs from spamming domains).

2. Some—like page ranking or other forms of blacklisting—rely on the source context; for example, all of the atoms in $\mathsf{get}(s)$ inherit the ranking assigned to the source s.[5]

3. Some—like authority—additionally rely on the structure of the inference rules in the original program. In this case, the quality and reliability of the meta-program itself is not questioned, where instead, the value of the annotation is derived (indirectly) from the T-atoms unified with the body of the rule. Accordingly, partially evaluated rules are assigned annotations based on the composition of the originating meta-rule and the provenance of the facts unified with the meta-rule body.

The first two kinds of annotation functions can be generalised as:

$$\psi_i : \mathsf{Facts} \times \mathsf{S} \to D_i$$

where, roughly speaking, they can be considered as a function of the corpus' quadruples.[6] We assume that $\psi_i(F, s)$ is defined for all $F \in \mathsf{get}(s)$. Slightly abusing notation, we use s_α to denote a special source containing the set of axiomatic triples, such that $\mathsf{get}s_\alpha := I_\alpha$: the set of axiomatic facts in the meta-program; we let $\psi_i(F, s_\alpha) := \top_i$ $(\forall F \in I_\alpha)$. Furthermore we assume without loss of generality that for some index z' $(0 \le z' \le z)$, $D_1, \ldots, D_{z'}$ are associated to annotation functions of this type.

[5] Strictly speaking, page ranking depends also on the hyperlinks occurring in context contents; these details are left implicit in our framework.

[6] We would also require information about redirects, but here we generalise.

Annotations of type 3 are produced from known information by functions of the form:

$$\psi_i : \mathsf{Rules} \times 2^{\mathsf{Facts}} \times \mathsf{S} \to D_i$$

where information about the rules is also required.[7] We assume that $\psi_i(R, \mathsf{get}(s), s)$ is defined for all $R \in P$; again, $\psi_i(R, I_\alpha, s_\alpha) := \top_i$. In addition, we define another special source s_τ for the terminological least model I_τ such that $\mathsf{get}(s_\tau) := I_\tau$ and where we assume $\psi_i(R, I_\tau, s_\tau)$ to be defined since the results of terminological reasoning may serve as (partial) instances of rules bodies.[8] As above, we assume without further loss of generality that $D_{z'+1}, \dots, D_z$ are associated to annotation functions of this type.[9]

6.3.5 T-split Annotated Programs

In order to integrate the annotated framework with our classical approach to reasoning—and following the discussion of the classical T-split least fixpoint in § 5.2—we now define a similar procedure for partially evaluating an annotated (meta-)program with respect to the terminological data to create an assertional annotated program; this procedure includes the aforementioned functions for deriving the seed annotations of the initial program.

Definition 6.9 (T-split annotated least fixpoint) *We define the* T-split annotated least fixpoint *as a two step process in analogy to the classical variant in Definition 5.3: (i) build and derive the least model of the terminological annotated program; (ii) build and derive the least model of the assertional annotated program. Starting with (i), let all rules (and facts) from the meta-program be annotated with* $\langle \top_1, \dots, \top_z \rangle$*. Next, let* $S \subset \mathsf{S}$ *denote the set of sources whose merged graphs comprise our corpus (including* s_α*); now, let:*

$$P^F := \bigcup_{s \in S} \left\{ F{:}\langle d_1, \dots, d_z \rangle \mid F \in \mathsf{get}(s), \underset{1 \le i \le z'}{\forall} \big(d_i = \psi_i(F, s)\big), \underset{z' < i \le z}{\forall} \big(d_i = \top_i\big) \right\}$$

denote the set of all annotated facts from the original corpus. We reuse $P^{T\emptyset}, P^{TA}, P^{\emptyset A}$ *as given in Definition 5.3 (each such meta-rule is now annotated with* $\langle \top_1, \dots, \top_z \rangle$*). Now, let* $TP := P^F \cup P^{T\emptyset}$ *denote the terminological annotated program, with least model* $\mathsf{lm}(TP)$*, and define the special source* s_τ *as before such that* $\mathsf{get}(s_\tau) := \mathsf{lm}(TP)$*. Next, let:*

$$\begin{aligned} P^{A+} := \bigcup_{s \in S \cup \{s_\tau\}} \big\{ \mathsf{Head}(R)\theta \leftarrow \mathsf{ABody}(R)\theta : \langle d_1, \dots, d_z \rangle \mid & R \in P^{TA}, \\ & \exists I \subseteq \mathsf{get}(s) \text{ s.t. } \theta = \mathsf{mgu}(\mathsf{TBody}(R), I), \\ & \langle d_1, \dots, d_{z'} \rangle = \mathsf{glb}(\{\mathbf{d'} \mid \exists F{:}\mathbf{d'} \in \mathsf{TBody}(R)\theta\}), \\ & \underset{z' < i \le z}{\forall} \big(d_i = \psi_i(R, \mathsf{TBody}(R)\theta, s)\big) \big\} \end{aligned}$$

denote the set of T-grounded rules with annotations up until z' *derived from the respective instances, and annotations thereafter derived from the annotation functions requiring knowledge about rules (e.g., authority).*[10] *Now, let* $AP := \mathsf{lm}(TP) \cup P^{\emptyset A} \cup P^{A+}$ *describe the assertional annotated program analogous to the classical version. Finally, we can give the* least model of the assertional program AP *as* $\mathsf{lm}(AP)$*—we more*

[7] One may note the correlation to the arguments of the authoritative T-grounding function in § 6.2.2.

[8] For authority, $\psi_{auth}(R, \mathsf{get}(s_\tau), s_\tau) = \bot_{auth}$ for all proper rules.

[9] As per the labelling in § 6.3.1, $|D_1|$ and $|D_3|$ (blacklisting and ranking respectively) are associated to the first form of labelling function, whereas $|D_2|$ (authority) is associated with the second form of labelling function—note that keeping the ranking domain in the last index allows for a more intuitive presentation of thresholding and finite domains in § 6.3.6.

[10] Note that this formulation of P^{A+} only allows residual rules to be created from T-instances which appear entirely in one source, as per discussion in § 6.2.2.

generally denote this by $\mathsf{lm}^T(P)$*: the* T-split least model *of the program* P*, where* AP *is derived from* P *as above.*

This definition of the T-split annotated least fixpoint is quite long, but follows the same intuition as for classical reasoning. First, all rules and facts in the meta-program are given the strongest possible truth value(s). Next, facts in the corpus are annotated according to functions which require only information about quadruples (annotations requiring knowledge of rules are annotated with $\top_i$). Then, the T-atom only rules are applied over the corpus (in particular, over the T-Box) and the terminological least model is generated (including annotations of the output facts). Next, rules with non-empty A-body *and* T-body have their T-atoms grounded with a set of T-facts, and the corresponding variable substitution is used to partially evaluate the A-body and head, where the resulting (proper) rule is annotated as follows: (i) the first category of annotation values ($1 \leq i \leq z'$) are given as the greatest lower bound of the truth values for the T-facts (thus, the rule is as "reliable" as the "least reliable" T-fact in the instance); (ii) the values for the second category of annotations ($z' < i \leq z$) are created as a function of the rule itself, the source of the T-facts, and the T-facts themselves. Finally, the assertional program is created, adding together the partially evaluated rules, the A-body only rules, and the facts in the terminological least fixpoint (which includes the set of original and axiomatic facts); the least model of this assertional program is then calculated to derive the T-split least model.

Note that during this process, rules and facts may be associated with more than one annotation tuple. Thus, although we will be applying the A-linear $\mathcal{O}2\mathcal{R}^{\propto\mathsf{A}}$ subset of OWL—and although the classical part of the program will still feature the same scalable properties as discussed in § 5.4.1—the growth of annotation tuples may still be polynomial (in our case cubic since $z = 3$; cf. Lemma 6.3) with respect to the set of original annotation values. In the next section, we look at scalability aspects of some reasoning tasks with respect to the annotated program (in particular, the assertional annotation program).

6.3.6 Annotated Reasoning Tasks

Within this framework, it is possible to define several types of reasoning tasks which, roughly speaking, refine the set of ground logical consequences according to optimality and threshold conditions. In this section, we introduce these reasoning tasks and look at the scalability of each in turn.

Plain: Returns all the ground logical consequences of P. Formally,

$$\mathsf{plain}(P) := \{F{:}\mathbf{d} \mid F \in \mathcal{B}_P \wedge P \models F{:}\mathbf{d}\}\,.$$

Optimal: Only the *non-dominated* elements of $\mathsf{plain}(P)$ are returned. Intuitively, an answer A—say, $\langle \mathbf{nb}, \mathbf{na}, 0.5\rangle$—can be ignored if a stronger evidence for A can be derived, for example $\langle \mathbf{nb}, \mathbf{a}, 0.6\rangle$. Formally, for an annotated program P (containing annotated rules and facts), let:

$$\mathsf{max}(P) := \{R{:}\mathbf{d} \in P \mid \forall R{:}\mathbf{d}' \in P(\mathbf{d} \not< \mathbf{d}')\}\,,$$

and define $\mathsf{opt}(P) = \mathsf{max}(\mathsf{plain}(P))$.

Above Threshold (*Optimal*): Refines $\mathsf{opt}(P)$ by selecting the optimal annotated consequences that are above a given threshold. Formally, given a threshold vector $\mathbf{t} \in \mathbf{D}$, let:

$$P_{\geq\mathbf{t}} := \{R{:}\mathbf{d} \in P \mid \mathbf{t} \leq \mathbf{d}\}\,,$$

and define $\mathsf{opt}_{\mathbf{t}}(P) = \mathsf{opt}(P)_{\geq\mathbf{t}}$.

Above Threshold (*Classical*): Returns the classical facts that have some annotation above a given threshold $\mathbf{t}$ such that annotations are not included in the answer. Formally, define:

$$\mathsf{above}_{\mathbf{t}}(P) := \{F \in \mathcal{B}_P \mid \exists \mathbf{d} \geq \mathbf{t}(\ P \models F{:}\mathbf{d})\}\,.$$

In our scenario, all tasks except *plain* enable us to prune the input program by dropping some facts/rules that do not contribute to the answers. For example, $\mathsf{opt}(P)$ does not depend on the dominated elements of P—when encountered, these elements can be discarded without affecting the task:

Theorem 6.8 $\mathsf{opt}(P) = \mathsf{opt}(\mathsf{max}(P))$.

Proof: Clearly, $\mathsf{max}(P) \subseteq P$ and both max and plain are monotonic with respect to set inclusion. Therefore, $\mathsf{plain}(\mathsf{max}(P))$ is contained in $\mathsf{plain}(P)$ and $\mathsf{max}(\mathsf{plain}(\mathsf{max}(P)))$ is contained in $\mathsf{max}(\mathsf{plain}(P))$; i.e., $\mathsf{opt}(\mathsf{max}(P)) \subseteq \mathsf{opt}(P)$.

For the opposite inclusion, we first prove by induction on natural numbers that for all $i \geq 0$ and $F \in \mathcal{B}_P$, if $\mathbf{d} \in \mathfrak{A}_P \uparrow i(F)$, then there exists an annotation $\mathbf{d}' \geq \mathbf{d}$ such that $\mathbf{d}' \in \mathfrak{A}_{\mathsf{max}(P)} \uparrow i(F)$.

The assertion is vacuously true for $i = 0$. Assume that $\mathbf{d} \in \mathfrak{A}_P \uparrow (i+1)(F)$, with $i \geq 0$, where we have that $\mathbf{d} = \mathsf{glb}(\{\mathbf{d}_1, \ldots, \mathbf{d}_n, \overline{\mathbf{d}}\})$ for some rule $F \leftarrow B_1, \ldots, B_n : \overline{\mathbf{d}}$ in $\mathsf{Ground}(P)$ and some associated $\mathbf{d}_1, \ldots, \mathbf{d}_n$; also, for all $1 \leq j \leq n$, we have that $\mathbf{d}_j \in \mathfrak{A}_P \uparrow i(B_j)$. By the definition of $\mathsf{max}(P)$, there exists a $F \leftarrow B_1, \ldots, B_n : \widehat{\mathbf{d}}$ in $\mathsf{Ground}(\mathsf{max}(P))$ with $\widehat{\mathbf{d}} \geq \overline{\mathbf{d}}$. Moreover, by induction, for all $1 \leq j \leq n$, there exists a $\mathbf{d}'_j \geq \mathbf{d}_j$ such that $\mathbf{d}'_j \in \mathfrak{A}_{\mathsf{max}(P)} \uparrow i(B_j)$. Thus if we set $\mathbf{d}' = \mathsf{glb}(\{\mathbf{d}'_1, \ldots, \mathbf{d}'_m, \widehat{\mathbf{d}}\})$, we have $\mathbf{d}' \geq \mathbf{d}$ and $\mathbf{d}' \in \mathfrak{A}_{\mathsf{max}(P)} \uparrow (i+1)(F)$.

Now assume that $F{:}\mathbf{d} \in \mathsf{opt}(P)$. In particular $F{:}\mathbf{d}$ is a logical consequence of P, therefore, by Lemma 6.6 and the previous statement, we have that there exists an annotation $F{:}\mathbf{d}' \in \mathsf{plain}(\mathsf{max}(P))$ with $\mathbf{d} \leq \mathbf{d}'$. However, since $\mathsf{opt}(\mathsf{max}(P)) \subseteq \mathsf{opt}(P)$, then $\mathsf{plain}(\mathsf{max}(P))$ cannot contain facts that improve $F{:}\mathbf{d}$, and therefore $\mathbf{d}' = \mathbf{d}$ and $F{:}\mathbf{d} \in \mathsf{opt}(\mathsf{max}(P))$. □

Similarly, if a minimal threshold is specified, then the program can be filtered by dropping all of the rules that are not above the given threshold:

Theorem 6.9 $\mathsf{opt}_{\mathbf{t}}(P) = \mathsf{opt}(P_{\geq \mathbf{t}})$.

Proof: Since by definition $\mathsf{opt}_{\mathbf{t}}(P) = \mathsf{max}(\mathsf{plain}(P))_{\geq \mathbf{t}}$ and also since $\mathsf{max}(\mathsf{plain}(P))_{\geq \mathbf{t}} = \mathsf{max}(\mathsf{plain}(P)_{\geq \mathbf{t}})$, it suffices to show that $\mathsf{plain}(P)_{\geq \mathbf{t}} = \mathsf{plain}(P_{\geq \mathbf{t}})$. By Lemma 6.6, $F{:}\mathbf{d} \in \mathsf{plain}(P)_{\geq \mathbf{t}}$ implies that for some i, $\mathbf{d} \in \mathfrak{A}_P \uparrow i(F)$ and $\mathbf{d} \geq \mathbf{t}$. Analogously to Theorem 6.8, it can be proven by induction on natural numbers that for all $i \geq 0$ and $F \in \mathcal{B}_P$, if $\mathbf{d} \in \mathfrak{A}_P \uparrow i(F)$ and $\mathbf{d} \geq \mathbf{t}$, then $\mathbf{d} \in \mathfrak{A}_{P_{\geq \mathbf{t}}} \uparrow i(F)$. Therefore, $\mathbf{d} \in \mathfrak{A}_{P_{\geq \mathbf{t}}} \uparrow i(F)$ and hence $F{:}\mathbf{d} \in \mathsf{plain}(P_{\geq \mathbf{t}})$.

Again, it can be proven by induction that if $F{:}\mathbf{d} \in \mathsf{plain}(P_{\geq \mathbf{t}})$, then $\mathbf{d} \geq \mathbf{t}$. Now, since plain is monotone with respect to set inclusion and $P_{\geq \mathbf{t}} \subseteq P$, then $\mathsf{plain}(P_{\geq \mathbf{t}}) \subseteq \mathsf{plain}(P)$; however, $\mathsf{plain}(P_{\geq \mathbf{t}})$ only contains annotations that dominate $\mathbf{t}$—hence $\mathsf{plain}(P_{\geq \mathbf{t}}) \subseteq \mathsf{plain}(P)_{\geq \mathbf{t}}$. □

Again, we will want to apply such reasoning tasks over corpora in the order of billions of facts sourced from millions of sources, so polynomial guarantees for the growth of annotations is still not sufficient for programs of this size—again, even quadratic growth may be too high. In order to again achieve our notion of *A-linearity* for the annotated case, for each $F \in \mathcal{B}_P$, the number of derived consequences and associated annotations $F{:}\mathbf{d}$ should remain linear with respect to the cardinality of P (assuming, of course, that P is A-linear in the classical sense). In the rest of this section, we assess the four reasoning tasks with respect to this requirement—in particular, we focus on the scalability of the *assertional* annotated program, where we accept the polynomial bound with respect to terminological knowledge, and again appeal to our assumption that the terminological segment of the corpus remains small.

From Lemma 6.3, we know that the cardinality of $\mathsf{plain}(P)$ is bounded by $|P|^z$; we can show this bound to be tight in the general case with an example:

Example 6.3 *Consider a z-dimensional $\mathbf{D}$ where each component i may assume an integer value from 1 to n. Let P be the following propositional program consisting of all rules of the following form:*

$$\begin{array}{lll} A_1 & : \langle m_1, n, \ldots, n\rangle & (1 \leq m_1 \leq n) \\ A_i \leftarrow A_{i-1} & : \langle n, \ldots, m_i, \ldots, n\rangle & (1 \leq m_i \leq n) \\ \ldots & & (\ 2 \leq i \leq z) \end{array}$$

where, intuitively, m_i assigns all possible values to each component i. Now, there are n facts which have every possible value for the first annotation component and the value n for all other components. Thereafter, for each of the remaining $z-1$ annotation components, there are n annotated rules which have every possible value for the given annotation component, and the value n for all other components. Altogether, the cardinality of P is nz. The set of annotations that can be derived for A_z is exactly $\mathbf{D}$, therefore its cardinality is n^z which grows as $\Theta(|P|^z)$. When $z \geq 2$, the number of labels associated to A_z alone exceeds the desired linear bound on materialisations.

To demonstrate this, let's instantiate P for $n = 2$ and $z = 3$:

$$\begin{array}{c} A_1 : \langle 1,2,2\rangle \ , \ A_1 : \langle 2,2,2\rangle \ , \\ A_2 \leftarrow A_1 : \langle 2,1,2\rangle \ , \ A_2 \leftarrow A_1 : \langle 2,2,2\rangle \ , \\ A_3 \leftarrow A_2 : \langle 2,2,1\rangle \ , \ A_3 \leftarrow A_2 : \langle 2,2,2\rangle \ . \end{array}$$

Here, $|P| = 2 * 3 = 6$. By $\mathsf{plain}(P)$, we will get:

$$\begin{array}{c} A_1 : \langle 1,2,2\rangle \ , \ A_1 : \langle 2,2,2\rangle \ , \\ A_2 : \langle 1,1,2\rangle \ , \ A_2 : \langle 1,2,2\rangle \ , \ A_2 : \langle 2,1,2\rangle \ , \ A_2 : \langle 2,2,2\rangle \ , \\ A_3 : \langle 1,1,1\rangle \ , \ A_3 : \langle 1,1,2\rangle \ , \ A_3 : \langle 1,2,1\rangle \ , \ A_3 : \langle 1,2,2\rangle \ , \\ A_3 : \langle 2,1,1\rangle \ , \ A_3 : \langle 2,1,2\rangle \ , \ A_3 : \langle 2,2,1\rangle \ , \ A_3 : \langle 2,2,2\rangle \ . \end{array}$$

Where A_3 is associated with $2^3 = 8$ annotations. ◊

This potential growth of $\mathsf{plain}(.)$ makes $\mathsf{opt}(.)$ a potentially appealing alternative; however, even if Theorem 6.8 enables some optimisation, in general $\mathsf{opt}(P)$ is not linear with respect to $|P|$. This can again be seen with an example:

Example 6.4 *Consider a program containing all rules of the form:*

$$\begin{array}{c} (\texttt{ex:Foo},\ \texttt{ex:spam},\ \texttt{ex:Bar}) : \langle k, \tfrac{1}{k}\rangle \\ \ldots \\ (\texttt{?x},\ \texttt{ex:spam\_}k,\ \texttt{?y}) \leftarrow (\texttt{?y},\ \texttt{ex:spam},\ \texttt{?x}) : \langle n, n\rangle \\ \ldots \end{array}$$

such that n is a (constant) positive integer and $1 \leq k \leq n$. By increasing n, P grows as $\Theta(2n)$, whereas (as per the previous example) $|\mathsf{plain}(P)|$ grows as $\Theta(n^2)$, with n facts of the form (`ex:Foo`, `ex:spam_`k, `ex:Bar`) being associated with n annotations of the form $\langle k, \frac{1}{k}\rangle$—thus, $|\mathsf{plain}(P)|$ grows quadratically with $|P|$. Now, for all such consequences relative to the same fact $F{:}\langle h, \frac{1}{h}\rangle$ and $F{:}\langle j, \frac{1}{j}\rangle$, if $h < j$, then $\frac{1}{j} < \frac{1}{h}$ and vice versa. This implies that all of the derived consequences are optimal—that is, $\mathsf{plain}(P) = \mathsf{opt}(P)$. Consequently, $|\mathsf{opt}(P)|$ grows quadratically with $|P|$, too.

To demonstrate this, let's instantiate P for $n = 3$:

$$\begin{gathered}
(\texttt{ex:Foo},\ \texttt{ex:spam},\ \texttt{ex:Bar}) : \langle 1,1\rangle\ , \\
(\texttt{ex:Foo},\ \texttt{ex:spam},\ \texttt{ex:Bar}) : \langle 2,\tfrac{1}{2}\rangle\ , \\
(\texttt{ex:Foo},\ \texttt{ex:spam},\ \texttt{ex:Bar}) : \langle 3,\tfrac{1}{3}\rangle\ , \\
(\texttt{?x},\ \texttt{ex:spam\_1},\ \texttt{?y}) \leftarrow (\texttt{?y},\ \texttt{ex:spam},\ \texttt{?x}) : \langle 3,3\rangle\ , \\
(\texttt{?x},\ \texttt{ex:spam\_2},\ \texttt{?y}) \leftarrow (\texttt{?y},\ \texttt{ex:spam},\ \texttt{?x}) : \langle 3,3\rangle\ , \\
(\texttt{?x},\ \texttt{ex:spam\_3},\ \texttt{?y}) \leftarrow (\texttt{?y},\ \texttt{ex:spam},\ \texttt{?x}) : \langle 3,3\rangle\ .
\end{gathered}$$

Here, $|P| = 2^3 = 6$. Now, $\mathsf{opt}(P)$*—or, equivalently,* $\mathsf{plain}(P)$*—will give $3^2 = 9$ annotated facts of the form:*

$$\left.\begin{gathered}
(\texttt{ex:Bar},\ \texttt{ex:spam\_}k,\ \texttt{ex:Foo}) : \langle 1,1\rangle\ , \\
(\texttt{ex:Bar},\ \texttt{ex:spam\_}k,\ \texttt{ex:Foo}) : \langle 2,\tfrac{1}{2}\rangle\ , \\
(\texttt{ex:Bar},\ \texttt{ex:spam\_}k,\ \texttt{ex:Foo}) : \langle 3,\tfrac{1}{3}\rangle\ .
\end{gathered}\ \right|\ \Big(\forall 1 \le k \le 3\Big)$$

Here, all 9 facts are considered optimal (as per Definition 6.5, we do not assume a lexicographical order). ◊

In this example, the annotations associated to each atom grow linearly with $|P|$. We can generalise this result and prove that the annotations of a given atom may grow as $|P|^{\frac{z}{2}}$ by slightly modifying Example 6.3 (which was for plain materialisation) along similar lines.

Example 6.5 *Assume that the dimension z is even, and that a component may assume any value from the set of rationals in the interval $[0, n]$; now, consider the program P containing all such facts and rules:*

$$\begin{array}{rl}
A_1 : \langle r_1, \frac{1}{r_1}, n \ldots, n\rangle & (1 \le r_1 \le n) \\
A_i \leftarrow A_{i-1} : \langle n, \ldots, n, r_{2i-1}, \frac{1}{r_{2i-1}}, n, \ldots, n\rangle & (1 \le r_{2i-1} \le n) \\
 & (2 \le i \le \frac{z}{2})
\end{array}$$

where r_{2i-1} assigns the $(2i-1)$th (odd) component all possible values between 1 and n inclusive, and $\frac{1}{r_{2i-1}}$ assigns the $2i$th (even) component all possible values between 1 and $\frac{1}{n}$ inclusive. Note that the cardinality of P is $\frac{nz}{2}$, containing n facts and $\frac{n(z-2)}{2}$ proper rules. Now, given two consequences $A{:}\mathbf{d}_1, A{:}\mathbf{d}_2 \in$ $\mathsf{plain}(P)$ *sharing the same atom A, there exist two distinct integers $j, k \le n$, $j \ne k$ and a pair of contiguous components $i, i+1 \le z$ such that $\mathbf{d}_1 = \langle \ldots, j, \frac{1}{j}, \ldots\rangle$ and $\mathbf{d}_2 = \langle \ldots, k, \frac{1}{k}, \ldots\rangle$. Therefore, all the facts in* $\mathsf{plain}(P)$ *are optimal, and the number of annotations for $A_{\frac{z}{2}}$ is $n^{\frac{z}{2}}$.*

To demonstrate this, let's instantiate P for $n = 2$ and $z = 6$:

$$\begin{gathered}
A_1 : \langle 1,1,2,2,2,2\rangle\ ,\ A_1 : \langle 2,\tfrac{1}{2},2,2,2,2\rangle\ , \\
A_2 \leftarrow A_1 : \langle 2,2,1,1,2,2\rangle\ ,\ A_2 \leftarrow A_1 : \langle 2,2,2,\tfrac{1}{2},2,2\rangle\ , \\
A_3 \leftarrow A_2 : \langle 2,2,2,2,1,1\rangle\ ,\ A_3 \leftarrow A_2 : \langle 2,2,2,2,2,\tfrac{1}{2}\rangle\ .
\end{gathered}$$

Here, $|P| = \frac{2\times 6}{2} = 6$. By $\mathsf{opt}(P)$*—or, equivalently, by plain(P)—we will get:*

$$\begin{gathered}
A_1 : \langle 1,1,2,2,2,2\rangle\ ,\ A_1 : \langle 2,\tfrac{1}{2},2,2,2,2\rangle\ , \\
A_2 : \langle 1,1,1,1,2,2\rangle\ ,\ A_2 : \langle 1,1,2,\tfrac{1}{2},2,2\rangle\ ,\ A_2 : \langle 2,\tfrac{1}{2},1,1,2,2\rangle\ ,\ A_2 : \langle 2,\tfrac{1}{2},2,\tfrac{1}{2},2,2\rangle\ , \\
A_3 : \langle 1,1,1,1,1,1\rangle\ ,\ A_3 : \langle 1,1,1,1,2,\tfrac{1}{2}\rangle\ ,\ A_3 : \langle 1,1,2,\tfrac{1}{2},1,1\rangle\ ,\ A_3 : \langle 1,1,2,\tfrac{1}{2},2,\tfrac{1}{2}\rangle\ , \\
A_3 : \langle 2,\tfrac{1}{2},1,1,1,1\rangle\ ,\ A_3 : \langle 2,\tfrac{1}{2},1,1,2,\tfrac{1}{2}\rangle\ ,\ A_3 : \langle 2,\tfrac{1}{2},2,\tfrac{1}{2},1,1\rangle\ ,\ A_3 : \langle 2,\tfrac{1}{2},2,\tfrac{1}{2},2,\tfrac{1}{2}\rangle\ .
\end{gathered}$$

Here, A_3 is associated with $2^{\frac{6}{2}} = 8$ optimal annotations.

It is not hard to adapt this example to odd z and prove that for all $z \in \mathbb{N}$, the number of annotations associated to an atom is in $\Theta(|P|^{\lfloor \frac{z}{2} \rfloor})$. Therefore, if the number of distinct atoms occurring in the answer can grow linearly (as in the aforementioned fragment $\mathcal{O}2\mathcal{R}^{\propto\mathsf{A}}$), then $|\mathsf{opt}(P)|$ *is in $\Theta(|P|^{\lfloor \frac{z}{2} \rfloor + 1})$ and hence not linear for $z > 1$.* ◊

If we further restrict our computations to the consequences above a given threshold (i.e., $\mathsf{opt}_{\mathbf{t}}(.)$), then some improvements may be possible (cf. Theorem 6.9). However, by setting $\mathbf{t}$ to the least element of $\mathbf{D}$, one can see that the worst case complexity of $\mathsf{opt}_{\mathbf{t}}(.)$ and $\mathsf{opt}(.)$ are the same.

The last reasoning task, $\mathsf{above}_{\mathbf{t}}(P)$, returns atoms without annotations, and hence it is less informative than the other tasks. However, it does not suffer from the performance drawbacks of the other tasks: $\mathsf{above}_{\mathbf{t}}(.)$ does not increase the complexity of annotation-free reasoning where it can be computed by dropping all rules (and facts) whose annotations are below $\mathbf{t}$ and reasoning classically with the remaining rules, as formalised by the following proposition:

Proposition 6.10 *Let* $\mathsf{lm}(P^{\mathbf{t}})$ *denote the least Herbrand model of the (classical) program* $P^{\mathbf{t}}$*. Then* $\mathsf{above}_{\mathbf{t}}(P) = \mathsf{lm}(P_{\geq\mathbf{t}})$ *.*

Proof: Let $\mathfrak{T}$ be the classical immediate consequence operator (as introduced in § 3.4) and define $\mathfrak{T}_{P_{\geq\mathbf{t}}} \uparrow \alpha$ by analogy with $\mathfrak{A}_{P_{\geq\mathbf{t}}} \uparrow \alpha$. It is straightforward to see by induction on natural numbers that for all $i \geq 0$, $\mathfrak{A}_{P_{\geq\mathbf{t}}} \uparrow i(F) \neq \emptyset$ iff $F \in \mathfrak{T}_{P_{\geq\mathbf{t}}} \uparrow i$. This means that $F \in \mathsf{lm}(P_{\geq\mathbf{t}})$ iff $\mathfrak{A}_{P_{\geq\mathbf{t}}} \uparrow \omega(F) \neq \emptyset$, or equivalently, iff for some $\mathbf{d}$, $F{:}\mathbf{d} \in \mathsf{plain}(P_{\geq\mathbf{t}})$. Moreover, as already shown in the proof of Theorem 6.9, $\mathsf{plain}(P_{\geq\mathbf{t}}) = \mathsf{plain}(P)_{\geq\mathbf{t}}$, therefore $F{:}\mathbf{d} \in \mathsf{plain}(P)_{\geq\mathbf{t}}$ (for some $\mathbf{d}$) iff $F \in \mathsf{above}_{\mathbf{t}}(P)$. □

The analysis carried out so far apparently suggests that $\mathsf{above}_{\mathbf{t}}$ is the only practically feasible inference among the four tasks. However, our use-case annotation domain—comprised of blacklisting, triple-rank and authoritativeness—enjoys certain properties that can be exploited to efficiently implement both opt and $\mathsf{opt}_{\mathbf{t}}$. Such properties bound the number of maximal labels with an expression that is constant with respect to P and depends only on the annotation domains: the most important property is that all D_is but one are finite (blacklisting and authoritativeness are boolean; only triple-ranks range over an infinite set of values). Thus, the number of maximal elements of any finite set of annotations is bounded by a linear function of $\mathbf{D}$.

Example 6.6 *Here we see an example of a fact with four non-dominated annotations.*

$$
\begin{aligned}
&(\texttt{ex:Foo}, \texttt{ex:precedes}, \texttt{ex:Bar}) : \langle \mathbf{b}, \mathbf{na}, 0.4 \rangle\,,\\
&(\texttt{ex:Foo}, \texttt{ex:precedes}, \texttt{ex:Bar}) : \langle \mathbf{b}, \mathbf{a}, 0.3 \rangle\,,\\
&(\texttt{ex:Foo}, \texttt{ex:precedes}, \texttt{ex:Bar}) : \langle \mathbf{nb}, \mathbf{na}, 0.3 \rangle\,,\\
&(\texttt{ex:Foo}, \texttt{ex:precedes}, \texttt{ex:Bar}) : \langle \mathbf{nb}, \mathbf{a}, 0.2 \rangle\,.
\end{aligned}
$$

Any additional annotation for this fact would either dominate or be dominated by a current annotation—in either case, the set of maximal annotations would maintain a cardinality of four. ◊

We now formalise and demonstrate this intuitive result. *For simplicity, we assume without further loss of generality that the finite domains are* $D_1, \ldots, D_{z-1}$*; we make no assumption on* D_z*.* (Note that for convenience, this indexing of the (in)finite domains replaces the former indexing presented in § 6.3.4 relating to how annotations are labelled—the two should be considered independent.)

First, with a little abuse of notation, given $\mathbf{D}' \subseteq \mathbf{D}$, $\mathsf{max}(\mathbf{D}')$ is the set of maximal values of $\mathbf{D}'$; formally, $\{\mathbf{d} \in \mathbf{D}' \mid \forall \mathbf{d}' \in \mathbf{D}'(\mathbf{d} \not< \mathbf{d}')\}$.

Theorem 6.11 *If* $D_1, \ldots, D_{z-1}$ *are finite, then for all finite* $\mathbf{D}' \subseteq \mathbf{D}$*,* $|\mathsf{max}(\mathbf{D}')| \leq \Pi_1^{z-1}|D_i|$ *.*

Proof: There exist at most $\Pi_1^{z-1}|D_i|$ combinations of the first $z-1$ components. Therefore, if $|\mathsf{max}(\mathbf{D}')|$ was greater than $\Pi_1^{z-1}|D_i|$, there would be two annotations $\mathbf{d}_1$ and $\mathbf{d}_2$ in $\mathsf{max}(\mathbf{D}')$ that differ only in the last component. But in this case either $\mathbf{d}_1 > \mathbf{d}_2$ or $\mathbf{d}_1 < \mathbf{d}_2$, and hence they cannot be both in $\mathsf{max}(\mathbf{D}')$ (a contradiction). □

As a consequence, in our reference scenario (where $z = 3$ and $|D_1| = |D_2| = 2$) each atom can be associated with at most 4 different maximal annotations. Therefore, if P is A-linear and if all but one domain are finite, then $\mathsf{opt}(P)$ is also A-linear.

However, a linear bound on the *output* of reasoning tasks does not imply the same bound on the intermediate steps (e.g., the alternative framework introduced in the next subsection needs to compute also non-maximal labels for a correct answer). Fortunately, a bottom-up computation that considers only maximal annotations is possible in this framework. Let $\mathfrak{A}_P^{\mathsf{max}}(I)$ be such that for all $F \in \mathcal{B}_P$: $\mathfrak{A}_P^{\mathsf{max}}(I)(A) = \mathsf{max}(\mathfrak{A}_P(I)(A))$, and define its powers $\mathfrak{A}_P^{\mathsf{max}} \uparrow \alpha$ by analogy with $\mathfrak{A}_P \uparrow \alpha$.

Lemma 6.12 *For all ordinals α, $\mathfrak{A}_P^{\mathsf{max}} \uparrow \alpha = \mathsf{max}(\mathfrak{A}_P \uparrow \alpha)$.*

Proof: First we prove the following claim (from which the lemma follows by induction):

$$\mathsf{max}(\mathfrak{A}_P(\mathsf{max}(I))) = \mathsf{max}(\mathfrak{A}_P(I)) .$$

The inclusion $\subseteq$ is trivial. For the other inclusion, assume that for some $H \in \mathcal{B}_P$, $\mathbf{d} \in \mathsf{max}(\mathfrak{A}_P(I)(H))$, this means that $\mathbf{d} \in \mathfrak{A}_P(I)(H)$ and for all $\mathbf{d}' \in \mathfrak{A}_P(I)(H)$, $\mathbf{d} \not< \mathbf{d}'$. As $\mathbf{d} \in \mathfrak{A}_P(I)(H)$, there exists a rule $H \leftarrow B_1, \ldots, B_n : \overline{\mathbf{d}}$ and some annotations $\mathbf{d}_1, \ldots, \mathbf{d}_n$ such that 1) $\mathbf{d} = \mathsf{glb}(\{\mathbf{d}_1, \ldots, \mathbf{d}_n, \overline{\mathbf{d}}\})$ and 2) for all $1 \leq j \leq n$, $\mathbf{d}_j \in I(B_j)$.

By definition, for all $1 \leq j \leq n$, there exists a $\mathbf{d}_j^{\mathsf{max}} \in \mathsf{max}(I)$ such that $\mathbf{d}_j \leq \mathbf{d}_j^{\mathsf{max}}$. Clearly, given

$$\mathbf{d}^{\mathsf{max}} = \mathsf{glb}(\{\mathbf{d}_1^{\mathsf{max}}, \ldots, \mathbf{d}_n^{\mathsf{max}}, \overline{\mathbf{d}}\}) ,$$

$\mathbf{d} \leq \mathbf{d}^{\mathsf{max}}$ and $\mathbf{d}^{\mathsf{max}} \in \mathfrak{A}_P(\mathsf{max}(I))$. However, since $\mathbf{d}$ is maximal with respect to $\mathfrak{A}_P(I)(H)$ and $\mathfrak{A}_P(\mathsf{max}(I))(H) \subseteq \mathfrak{A}_P(I)(H)$, then $\mathbf{d} \leq \mathbf{d}^{\mathsf{max}}$ and $\mathbf{d} \in \mathsf{max}(\mathfrak{A}_P(\mathsf{max}(I)))$. This proves the claim—thereafter, the lemma follows by a straightforward induction. □

Simply put, we only need knowledge of maximal annotations to compute maximal annotations by the glb function—dominated annotations are redundant and can be removed at each step.

Although $\mathfrak{A}_P^{\mathsf{max}}$ is not monotonic—annotations may be replaced by new maximal annotations at later steps[11]—it follows from Lemma 6.12 that $\mathfrak{A}_P^{\mathsf{max}}$ reaches a fixpoint; further, when P is Datalog, this fixpoint is reached in a finite number of steps:

Theorem 6.13 *If P is Datalog, then there exists $i < \omega$ such that*

1. *$\mathfrak{A}_P^{\mathsf{max}} \uparrow i$ is a fixpoint of $\mathfrak{A}_P^{\mathsf{max}}$;*
2. *$\mathfrak{A}_P^{\mathsf{max}} \uparrow j$ is not a fixpoint of $\mathfrak{A}_P^{\mathsf{max}}$, for all $0 \leq j < i$;*
3. *$F{:}\mathbf{d} \in \mathsf{opt}(P)$ iff $\mathbf{d} \in \mathfrak{A}_P^{\mathsf{max}} \uparrow i(F)$.*

Proof: If P is Datalog, by Lemma 6.7, for some $k < \omega$, $\mathfrak{A}_P \uparrow k = \mathsf{lfp}(\mathfrak{A}_P)$; we will now show that $\mathfrak{A}_P^{\mathsf{max}} \uparrow k$ is a fixpoint as well. By definition

$$\mathfrak{A}_P^{\mathsf{max}}(\mathfrak{A}_P^{\mathsf{max}} \uparrow k) = \mathsf{max}(\mathfrak{A}_P(\mathfrak{A}_P^{\mathsf{max}} \uparrow k)) ,$$

by Lemma 6.12, $\mathfrak{A}_P^{\mathsf{max}} \uparrow k = \mathsf{max}(\mathfrak{A}_P \uparrow k)$, so we have

$$\mathfrak{A}_P^{\mathsf{max}}(\mathfrak{A}_P^{\mathsf{max}} \uparrow k) = \mathsf{max}(\mathfrak{A}_P(\mathsf{max}(\mathfrak{A}_P \uparrow k))) .$$

[11] One may consider the end of Example 6.3, where instead of applying $\mathsf{lfp}(\mathfrak{A}_P)$ if one applies $\mathsf{lfp}(\mathfrak{A}_P^{\mathsf{max}})$, then all but the $\langle 2, 2, 2\rangle$ annotation value for A_1, A_2 and A_3 are eventually dominated and thus removed.

However, as already shown in the proof of Lemma 6.12, for any I, $\mathsf{max}(\mathfrak{A}_P(\mathsf{max}(I))) = \mathsf{max}(\mathfrak{A}_P(I))$. Therefore,

$$\mathfrak{A}_P^{\mathsf{max}}(\mathfrak{A}_P^{\mathsf{max}} \uparrow k) = \mathsf{max}(\mathfrak{A}_P(\mathfrak{A}_P \uparrow k))\,.$$

Finally, since $\mathfrak{A}_P \uparrow k$ is a fixpoint and reusing Lemma 6.12

$$\mathfrak{A}_P^{\mathsf{max}}(\mathfrak{A}_P^{\mathsf{max}} \uparrow k) = \mathsf{max}(\mathfrak{A}_P \uparrow k) = \mathfrak{A}_P^{\mathsf{max}} \uparrow k\,.$$

Thus, $\mathfrak{A}_P^{\mathsf{max}} \uparrow k$ is a fixpoint and hence, by finite regression, there exists an $0 \leq i \leq k$ such that $\mathfrak{A}_P^{\mathsf{max}} \uparrow i$ is a fixpoint, where for all $0 \leq j < i$, $\mathfrak{A}_P^{\mathsf{max}} \uparrow j$ is not a fixpoint.

Clearly, $\mathfrak{A}_P^{\mathsf{max}} \uparrow k = \mathfrak{A}_P^{\mathsf{max}} \uparrow i$. Since $\mathfrak{A}_P^{\mathsf{max}} \uparrow k = \mathsf{max}(\mathfrak{A}_P \uparrow k)$ (Lemma 6.12), we finally have

$$\mathfrak{A}_P^{\mathsf{max}} \uparrow i = \mathsf{max}(\mathsf{lfp}(\mathfrak{A}_P))\,.$$

Therefore, $\mathbf{d} \in \mathfrak{A}_P^{\mathsf{max}} \uparrow i$ iff $F{:}\mathbf{d} \in \mathsf{max}(\mathsf{plain}(P)) = \mathsf{opt}(P)$. □

Loosely, since the fixpoint of $\mathfrak{A}_P^{\mathsf{max}}$ can also be reached by deriving the (known) finite fixpoint of $\mathfrak{A}_P$ and thereafter removing the dominated annotations (which is known to be equivalent to removing the dominated annotations at each step) $\mathfrak{A}_P^{\mathsf{max}}$ also has a finite fixpoint.

Theorem 6.11 ensures that at every step j, $\mathfrak{A}_P^{\mathsf{max}} \uparrow j$ associates each derived atom to a constant maximum number of annotations that is independent of $|P|$. By Theorem 6.9, the bottom-up construction based on $\mathfrak{A}_P^{\mathsf{max}}$ can be used also to compute $\mathsf{opt}_{\mathbf{t}}(P) = \mathsf{opt}(P_{\geq \mathbf{t}})$. Informally speaking, this means that if $D_1, \ldots, D_{z-1}$ are finite, then both $\mathsf{opt}(.)$ and $\mathsf{opt}_{\mathbf{t}}(.)$ are feasible.

Furthermore, our typical use-case will be to derive optimal non-blacklisted and authoritative inferences; we can formalise this as the task $\mathsf{opt}_{\mathbf{t}}(.)$ with a threshold $\mathbf{t}$ which has $z-1$ components set to their maximum possible value, such that each atom can be associated to one optimal annotation (in our use-case, the highest triple-rank value). *For the sake of simplicity, we assume without loss of generality that the threshold elements set to the maximum value are the first $z-1$.*

Lemma 6.14 *Let $\mathbf{t} = \langle t_1, \ldots, t_z \rangle$. If $t_i = \top_i$ for $1 \leq i < z$, then for all $\mathbf{D}' \subseteq \mathbf{D}$, $|\mathsf{max}(\mathbf{D}'_{\geq \mathbf{t}})| \leq 1$.*

Proof: If not empty, all of the annotations in $\mathbf{D}'_{\geq \mathbf{t}}$ are of type $\langle \top_1, \ldots, \top_{z-1}, d_z \rangle$, thus max selects the one with the maximal value of d_z. □

As a consequence, each atom occurring in $\mathsf{opt}_{\mathbf{t}}(P)$ is associated to one annotation, and the same holds for the intermediate steps $\mathfrak{A}_{P_{\geq \mathbf{t}}}^{\mathsf{max}} \uparrow j$ of the iterative construction of $\mathsf{opt}_{\mathbf{t}}(P)$:

Theorem 6.15 *Assume that P is Datalog and that the threshold assumption of Lemma 6.14 again holds. Let i be the least index such that $\mathfrak{A}_{P_{\geq \mathbf{t}}}^{\mathsf{max}} \uparrow i$ is a fixpoint of $\mathfrak{A}_{P_{\geq \mathbf{t}}}^{\mathsf{max}}$. Then*

1. *if $\{F{:}\mathbf{d}_1, F{:}\mathbf{d}_2\} \subseteq \mathsf{opt}_{\mathbf{t}}(P)$, then $\mathbf{d}_1 = \mathbf{d}_2$;*
2. *if $\{\mathbf{d}_1, \mathbf{d}_2\} \subseteq \mathfrak{A}_{P_{\geq \mathbf{t}}}^{\mathsf{max}} \uparrow j(F)$ $(0 \leq j \leq i)$, then $\mathbf{d}_1 = \mathbf{d}_2$.*

Proof: We focus on proving the second assertion (from which the first follows naturally). For $j = 0$, the assertion is vacuously true. For $j > 0$, $\mathfrak{A}_{P_{\geq \mathbf{t}}}^{\mathsf{max}} \uparrow j(F) = \mathsf{max}(\mathfrak{A}_{P_{\geq \mathbf{t}}}(\mathfrak{A}_{P_{\geq \mathbf{t}}}^{\mathsf{max}} \uparrow (j-1))(F))$, therefore both $\mathbf{d}_1$ and $\mathbf{d}_2$ are maximal ($\mathbf{d}_1 \not< \mathbf{d}_2$ and $\mathbf{d}_1 \not> \mathbf{d}_2$). But $\mathbf{d}_1$ and $\mathbf{d}_2$ differ only for the last component z, and since $\leq_z$ is a total order, then $\mathbf{d}_1 = \mathbf{d}_2$. □

The annotated reasoning experiments we will conduct over our corpus belong to this case where, formally, we define our threshold as

$$\mathbf{t} := \langle \top_1, \top_2, \bot_3 \rangle = \langle \mathbf{nb}, \mathbf{a}, 0 \rangle\,,$$

where we derive only non-blacklisted, authoritative inferences, but with any triple-rank value. Accordingly, the implementation maintains (at most) a single annotation for each rule/fact at each stage; thus, $\mathsf{opt}_{\mathsf{t}}(P)$ for this threshold has minimal effect on the scalable properties of our classical reasoning algorithm.

An alternative approach

The discussion above shows that, in the general case, scalability problems may arise from the existence of a polynomial number of maximal annotations for the same atom. Then it may be tempting to force a total order on annotations and keep for each atom only its (unique) best annotation, in an attempt to obtain a complexity similar to above-threshold reasoning. In our reference scenario, for example, it would make sense to order annotation triples lexicographically, thereby giving maximal importance to blacklisting, medium importance to authoritativeness, and minimal importance to page ranking, so that—for example—$\langle \mathbf{nb}, \mathbf{na}, 0.9\rangle \leq \langle \mathbf{nb}, \mathbf{a}, 0.8\rangle$. Then interpretations could be restricted by forcing $I(F)$ to be always a singleton, containing the unique maximal annotation for F according to the lexicographic ordering.

Unfortunately, this idea does not work well together with the standard notion of rule satisfaction introduced before. In general, in order to infer the correct maximal annotation associated to a fact F it may be necessary to keep some non-maximal annotation, too (therefore the analogue of Lemma 6.12 does not hold in this setting).

Example 6.7 *Consider for example the program:*

$$\begin{aligned} H \leftarrow B &: \langle \mathbf{nb}, \mathbf{na}, 1.0\rangle \\ B \quad &: \langle \mathbf{nb}, \mathbf{na}, 0.9\rangle \\ B \quad &: \langle \mathbf{nb}, \mathbf{a}, 0.8\rangle \,. \end{aligned}$$

The best proof of H makes use of the first two rules/facts of the program, and gives H the annotation $\langle \mathbf{nb}, \mathbf{na}, 0.9\rangle$ since none of these rules/facts are blacklisted or authoritative, and the least triple-rank is 0.9. However, if we could associate each atom to its best annotation only, then B would be associated to $\langle \mathbf{nb}, \mathbf{a}, 0.8\rangle$, and the corresponding label for H would necessarily be the non-maximal $\langle \mathbf{nb}, \mathbf{na}, 0.8\rangle$ by the definition of rule satisfaction; therefore these semantics (in conjunction with lexicographic ordering) do not faithfully reflect the properties of the best proof of H. ◊

Currently we do not know whether any alternative, reasonable semantics can solve this problem, and we leave this issue as an open question—in any case, we note that this discussion does not affect our intended use-case of deriving $\mathsf{opt}_{\mathsf{t}}(P)$ for our threshold since, as per Theorem 6.15, we need only consider the total ordering given by the triple ranks.

6.3.7 Constraints

Our demonstrative use-case for annotations is to compute strengths of derivations such that can be used for repairing inconsistencies (a.k.a. contradictions) in a non-trivial manner, where we view inconsistency as the consequence of unintended publishing errors or unforeseen inferencing.[12] Thus, in order to detect inconsistencies, we require a special type of rule, which we call a *constraint*.

A constraint is a rule without a head, like:

$$\leftarrow A_1, \ldots, A_n, T_1, \ldots, T_m \quad (n, m \geq 0) \tag{6.2}$$

[12] We do however acknowledge the possibility of deliberate inconsistency, although we informally claim that such overt disagreement is not yet prevalent in Linked Data.

where $T_1, \ldots, T_m$ are T-atoms and $A_1, \ldots, A_n$ are A-atoms. As before, let

$$\mathsf{Body}(R) := \{A_1, \ldots, A_n, T_1, \ldots, T_m\}\,,$$

and let

$$\mathsf{TBody}(R) := \{T_1, \ldots, T_m\}\,.$$

We interpret such rules as indicators that instances of the body are inconsistent in and of themselves—as such, they correspond to a number of OWL 2 RL/RDF rules who have the special propositional symbol `false` to indicate contradiction (we leave the `false` implicit).

Example 6.8 *Take the OWL 2 RL/RDF meta-constraint* $C_{\mathsf{cax-dw}}$

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

where $\mathsf{TBody}(C_{\mathsf{cax-dw}})$ *is again underlined. Any instance of the body of this rule denotes an inconsistency.*◊

Classical semantics prescribes that a Herbrand model I *satisfies* a constraint C iff I satisfies *no* instance of $\mathsf{Body}(C)$. Consequently, if P is a logic program with constraints, either the least model of P's rules satisfies all constraints in P or P is inconsistent (in this case, under classical semantics, no reasoning can be carried out with P).

Annotations create an opportunity for a more flexible and reasonable use of constraints for a corpus collected from unvetted sources. Threshold-based reasoning tasks can be used to ignore the consequences of constraint violations based on low-quality or otherwise unreliable proofs. In the following, let $P = P^R \cup P^C$, where P^R is a set of rules and P^C a set of constraints.

Definition 6.10 (Threshold-consistent) *Let* $\mathbf{t} \in \mathbf{D}$. P *is* $\mathbf{t}$*-consistent iff* $\mathsf{above}_{\mathbf{t}}(P^R)$ *satisfies all the constraints of* P^C .

For example, if $\mathbf{t} = \langle \mathbf{nb}, \mathbf{a}, 0\rangle$ and P is $\mathbf{t}$-consistent, then for all constraints $C \in P^C$ all the proofs of $\mathsf{Body}(C)$ use either blacklisted facts or non-authoritative rules. This form of consistency can be equivalently characterised in terms of the alternative threshold task $\mathsf{opt}_{\mathbf{t}}$ which, unlike $\mathsf{above}_{\mathbf{t}}$, will generate annotated consequences; for all sets of annotated rules and facts Q, let $[Q] = \{R \mid R : \mathbf{d} \in Q\}$. Then we have:

Proposition 6.16 P *is* $\mathbf{t}$*-consistent iff* $[\mathsf{opt}_{\mathbf{t}}(P^R)]$ *satisfies all the constraints in* P^R.

More generally, the following definitions can be adopted for measuring the *strength* of constraint violations:

Definition 6.11 (Answers) *Let* $G = \{A_1, \ldots, A_n\}$ *be a set of atoms and let* $P = P^R \cup P^C$. *An* answer *for* G *(from* P*) is a pair* $\langle \theta, \mathbf{d}\rangle$ *where* θ *is a grounding substitution and*

1. *there exist* $\mathbf{d}_1, \ldots, \mathbf{d}_n \in \mathbf{D}$ *such that* $P^R \models A_i\theta{:}\mathbf{d}_i$,
2. $\mathbf{d} = \mathsf{glb}\{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$.

The set of all answers of G *from* P *is denoted by* $\mathsf{Ans}_P(G)$.[13]

Definition 6.12 (Annotated constraints, violation degree) *We define* annotated constraints *as expressions* $C{:}\mathbf{d}$ *where* C *is a constraint and* $\mathbf{d} \in \mathbf{D}$. *The* violation degree *of* $C{:}\mathbf{d}$ *wrt. program* P *is the set:*

$$\mathsf{max}(\{\mathsf{glb}(\mathbf{d}, \mathbf{d}') \mid \langle \theta, \mathbf{d}'\rangle \in \mathsf{Ans}_P(\mathsf{Body}(C))\})\,.$$

[13] One may note a correspondence to conjunctive-query answering, or in the case of RDF, basic-graph pattern matching—herein however, answers are additionally associated with an annotation, computed as the glb of annotations for facts contributing to the answer.

Intuitively, violation degrees provide a way of assessing the severity of inconsistencies by associating each constraint with the rankings of their strongest violations. Note further that T-split constraints will be partially evaluated and annotated alongside and analogously to other rules.

Example 6.9 *From Example 6.8, take the annotated OWL 2 RL/RDF meta-constraint $C_{\mathsf{cax-dw}}$:*

$$\leftarrow \underline{(\texttt{?c1}, \texttt{owl:disjointWith}, \texttt{?c2})}, (\texttt{?x}, \texttt{a}, \texttt{?c1}),(\texttt{?x}, \texttt{a}, \texttt{?c2}) : \langle \top_{\mathbf{1}}, \top_{\mathbf{2}}, \top_{\mathbf{3}} \rangle$$

and consider the following annotated facts:

$$\begin{aligned}
(\texttt{foaf:Organization}, \texttt{owl:disjointWith}, \texttt{foaf:Person}) &: \langle \mathbf{nb}, \mathbf{a}, 0.6 \rangle \\
(\texttt{ex:W3C}, \texttt{a}, \texttt{foaf:Organization}) &: \langle \mathbf{nb}, \mathbf{a}, 0.4 \rangle \\
(\texttt{ex:W3C}, \texttt{a}, \texttt{foaf:Person}) &: \langle \mathbf{nb}, \mathbf{na}, 0.3 \rangle \\
(\texttt{ex:TimBL}, \texttt{a}, \texttt{foaf:Organization}) &: \langle \mathbf{b}, \mathbf{na}, 0.8 \rangle \\
(\texttt{ex:TimBL}, \texttt{a}, \texttt{foaf:Person}) &: \langle \mathbf{nb}, \mathbf{a}, 0.7 \rangle
\end{aligned}$$

(Note that with respect to authority annotation values, in practice, the non-authoritative facts above can only be inferred through a non-authoritative rule.)

During the creation of the assertional program (during which rules are partially evaluated with respect to terminological knowledge) the rule $C_{\mathsf{cax-dw}}$, together with the first T-fact above, will generate the assertional annotated constraint:

$$\leftarrow (\texttt{?x}, \texttt{a}, \texttt{foaf:Organization}),(\texttt{?x}, \texttt{a}, \texttt{foaf:Person}) : \langle \mathbf{nb}, \mathbf{a}, 0.6 \rangle$$

(Note further that for T-ground constraints, we use the same authoritative function for annotation labelling, where the above rule can only be authoritative if the source providing the first fact is redirected to by either `foaf:Person` *or* `foaf:Organization`*—the terms substituted for variables appearing in both* ABody *and* TBody*.)*

This assertional rule has two answers *from the original facts:*

$$\big(\{\texttt{?x/ex:W3C}\}\ , \langle \mathbf{nb}, \mathbf{na}, 0.3 \rangle\big)$$
$$\big(\{\texttt{?x/ex:TimBL}\}\ , \langle \mathbf{b}, \mathbf{na}, 0.6 \rangle\big)$$

The violation degree of $C_{\mathsf{cax-dw}}$ is then $\{\langle \mathbf{nb}, \mathbf{na}, 0.3 \rangle, \langle \mathbf{b}, \mathbf{na}, 0.6 \rangle\}$ since neither annotation dominates the other. ◊

The computation of violation degrees can be reduced to opt by means of a simple program transformation. Suppose that $P^C = \{C_1{:}\mathbf{d}_1, \ldots, C_n{:}\mathbf{d}_n\}$. Introduce a fresh propositional symbol f_i for each C_i (i.e., in the case of OWL 2 RL/RDF, a symbol representing `false` specifically for each constraint) and let

$$P' = P^R \cup \{f_i \leftarrow \mathsf{Body}(C_i) : \mathbf{d}_i \mid i = 1, \ldots, n\}.$$

Proposition 6.17 *An annotation* $\mathbf{d}$ *belongs to the violation degree of $C_i{:}\mathbf{d}_i$ iff $f_i{:}\mathbf{d} \in \mathsf{opt}(P')$.*

The computation of violation degrees and thresholds can be combined, picking up only those annotations that are above threshold. This can be done by selecting all $\mathbf{d}$ such that $f_i{:}\mathbf{d} \in \mathsf{opt}_{\mathbf{t}}(P')$—as such, in our use-case we will again only be looking for violations above our threshold $\mathbf{t} := \langle \top_1, \top_2, \bot_3 \rangle$.

Now, one could consider identifying the threshold $\mathbf{t}$ such that the program P^R is $\mathbf{t}$-consistent and setting that as the new threshold to ensure consistency; however, in practice, this may involve removing a large, consistent portion of the program. In fact, in our empirical analysis (which will be presented later in § 6.4.3), we found that the 23rd overall ranked document in our corpus was already inconsistent due to an invalid datatype; thus, applying this brute-force threshold method would leave us with (approximately) a corpus of 22 documents, with almost four million sources being discarded. Instead, we use our annotations for a more granular repair of the corpus, where, roughly speaking, if $C_i : \mathbf{d}_i$ is violated, then the members of $\mathsf{Body}(C_i)\theta$ with the weakest proof are good candidates for deletion. We will sketch such a repair process in § 6.4.3.

6.4 Annotated Linked Data Reasoning

In this section, we move towards applying the presented methods for annotated reasoning over our evaluation Linked Data corpus of 1.118b quads, describing our distributed implementation, sketching a process for repairing inconsistencies, and ultimately reporting on our evaluation. In particular:

1. we begin by briefly describing our implementation of the triple-ranking procedure (§ 6.4.1);
2. along the same lines as the classical reasoning implementation, we detail our distributed methods for applying annotated reasoning (§ 6.4.2);
3. we sketch our algorithms for repairing inconsistencies using the annotations attached to facts (§ 6.4.3).

Each subsection concludes with evaluation over our Linked Data corpus.

6.4.1 Ranking Triples: Implementation/Evaluation

In this section, we briefly describe and evaluate our distributed methods for applying a PageRank-inspired analysis of the data to derive rank annotations for facts in the corpus based on the source ranks calculated in § 4.3, using the summation thereof outlined in § 6.2.3.

Distributed Ranking Implementation

Assuming that the sources ranks are available as a sorted file of context rank pairs of the form (c, r) (§ 4.3), and that the input data are available pre-distributed over the slave machines and sorted by context (as per § 4.3), the distributed procedure (as per the framework in § 3.6) for calculating the triple ranks is fairly straightforward:

1. **run**: each slave machine performs a merge-join over the ranks and its segment of the data (sorted by context), propagating ranks of contexts to individual quadruples and outputing quintuples of the form (s, p, o, c, r)—the ranked data are subsequently sorted by natural lexicographical order;
2. **coordinate**: each slave machine splits its segment of sorted ranked data by a hash-function (modulo slave-machine count) on the subject position, with split fragments sent to a target slave machine; each slave machine concurrently receives and stores split fragments from its peers;
3. **run**: each slave machine merge-sorts the subject-hashed fragments it received from its peers, summating the ranks for triples which appear in multiple contexts while streaming the data.

The result on each slave machine is a flat file of sorted quintuples of the form (s, p, o, c, r), where c denotes context and r rank, and where $r_i = r_j$ if $(s_i, p_i, o_i) = (s_j, p_j, o_j)$.

Ranking Evaluation and Results

Firstly, from § 4.3, the source-level ranking takes 30.3 h. Thereafter, deriving the triple ranks takes 4.2 h, with the bulk of time consumed as follows: (i) propagating source ranks to triples and hashing/coordinating and sorting the initial ranked data by subject took 3 h; (ii) merge-sorting and aggregating the ranks for triples took 1.2 h.

6.4.2 Reasoning: Implementation/Evaluation

Our annotated reasoning implementation closely follows the distributed classical reasoning approach (presented in § 5.4.1) where we again apply $\mathcal{O}2\mathcal{R}^{\propto\mathsf{A}}$ (§ 5.4.1): the A-linear subset of OWL 2 RL/RDF (listed in Appendix B). Herein, we first recap the high-level distributed reasoning steps—discussing amendments required for annotated reasoning—and subsequently present evaluation.

Distributed Implementation

As per § 6.3.6, we assume the threshold:

$$\mathbf{t} := \langle \top_1, \top_2, \bot_3 \rangle = \langle \mathbf{nb}, \mathbf{a}, 0 \rangle$$

for our experiments. Thus, our reasoning task becomes $\mathsf{opt}_{\mathbf{t}}(P)$. By Theorem 6.8, we can filter dominated facts/rules; by Theorem 6.9, we can filter blacklisted or non-authoritative facts/rules when they are first encountered.

Again, currently, we do not use the blacklisting annotation. In any case, assuming a threshold for non-blacklisted annotations, blacklisted rules/facts in the program could simply be filtered in a pre-processing step.

For the purposes of the authoritativeness annotation—and as per classical reasoning—the authority of individual terms in ground T-atoms are computed during the T-Box extraction phase. This intermediary *term-level authority* is then used by the master machine to annotate T-ground rules with the final authoritative annotation. Recall that all initial ground facts and proper rules $\mathcal{O}2\mathcal{R}^{\propto\mathsf{A}}$ are annotated as authoritative, and that only T-ground rules with non-empty ABody can be annotated as **na**, and subsequently that ground atoms can only be annotated with **na** if produced by such a non-authoritative rule; thus, with respect to our threshold $\mathbf{t}$, we can immediately filter any T-ground rules annotated **na** from the assertional program—thereafter, only **a** annotations can be encountered.

Thus, in practice, once blacklisted and non-authoritative facts/rules have been removed from the program, we need only maintain ranking annotations: in fact, following the discussion of § 6.3.6 and the aforementioned thresholding, we can extend the classical reasoning implementation in a straightforward manner.

As input, all $\mathcal{O}2\mathcal{R}^{\propto\mathsf{A}}$ axiomatic triples and meta-rules are annotated with $\top$ (the value 1). All other ground facts in the corpus are pre-assigned a rank annotation by the links-based ranking procedure—we assume that data (in the form of ranked quintuples containing the RDF triple, context, and rank) are resident on the slave machines, and are hashed by subject and sorted by lexicographical order (as is the direct result of the distributed ranking procedure in § 6.4.1). We can then apply the following distributed approach to our annotated reasoning task, where we use '*' to highlight tasks which are not required in the classical (non-annotated) reasoning implementation of § 5.4.3:

1. **run/gather:** identify and separate out the annotated T-Box from the main corpus in parallel on the slave machines, and subsequently merge the T-Box on the master machine;

2. **run:** apply axiomatic and "T-Box only" rules on the master machine, ground the T-atoms in rules with

non-empty T-body/A-body (throwing away non-authoritative rules and dominated rank annotation values) and build the linked annotated-rule index;

3. **flood/run:** send the linked annotated rule index to all slave machines, and reason over the main corpus in parallel on each machine, producing an on-disk output of rank annotated facts;
4. **coordinate:*** redistribute the inferred data on the slave machines by hashing on subject, including the T-Box reasoning results hitherto resident on the master machine;
5. **run:*** perform a parallel sort of the hashed/inferred data segments;
6. **run:*** filter any dominated facts using an on-disk merge-join of the sorted and ranked input and inferred corpora, streaming the final output.

The first step involves extracting and reasoning over the rank-annotated T-Box. Terminological data are extracted in parallel on the slave machines from the ranked corpus. These data are gathered onto the master machine. Ranked axiomatic and terminological facts are used for annotated T-Box level reasoning: internally, we store annotations using a map (alongside the triple store itself), and the semi-naïve evaluation only considers unique or non-dominated annotated facts in the delta. Inferred annotations are computed using the glb function as described in Definition 6.7.

Next, rules with non-empty ABody are T-ground. If TBody is not empty, the T-ground rule annotation is again given by the glb aggregation of the T-ground instance annotations; otherwise, the annotation remains $\top$. The master machine creates a linked rule index for the assertional program and floods it to all slave machines, who then begin reasoning.

Since our T-ground rules now only contain a single A-atom (as per the definition of $\mathcal{O}2\mathcal{R}^{\propto\mathsf{A}}$, § 5.4.1), during assertional reasoning, the glb function takes the annotation of the instance of the A-atom and the annotation of the T-ground rule to produce the inferred annotation. For the purposes of duplicate removal, our LRU cache again considers facts with dominated annotations as duplicate.

Finally, since our assertional reasoning procedure is not semi-naïve and can only perform partial duplicate detection (as per § 5.2.1), we may have duplicates or dominated facts in the output (both locally and globally). To ensure optimal output—and thus achieve $\mathsf{opt}_\mathsf{t}(P)$—we must finally apply the last three steps (marked with an asterisk).[14] We must split and coordinate the inferred data (by subject) across the slave machines and subsequently sort these data in parallel—note that the input data are already split and sorted during ranking, and we assume that the same split function and sorting is used for the inferred data. Subsequently, each slave machine scans and merges the input and inferred data; during the scan, for removing dominated annotations, each individual unique triple is kept in memory along with its highest annotation value, which are output when the next triple group (or the end of the file) is encountered.

Reasoning Evaluation

We again apply our methods over nine machines: eight slave machines and one master machine. Note that we do not merge rules since we want to avoid associating different atoms in the head with different annotations, and we do not saturate rules. However, we do remove equivalent/dominated rules, and we again maintain a linked rule index (as per § 5.3).

In total, the outlined distributed reasoning procedure—including aggregation of the final results—took 14.6 h. The bulk of time was consumed as follows: (i) extracting the T-Box in parallel took 51 min; (ii) gathering the T-Box locally onto the master machine took 2 min; (iii) locally reasoning over the T-Box took

[14] One could use a similar approach for removing duplicates for the classical reasoning approach; similarly, one could consider duplicate removal as unnecessary for certain use-cases.

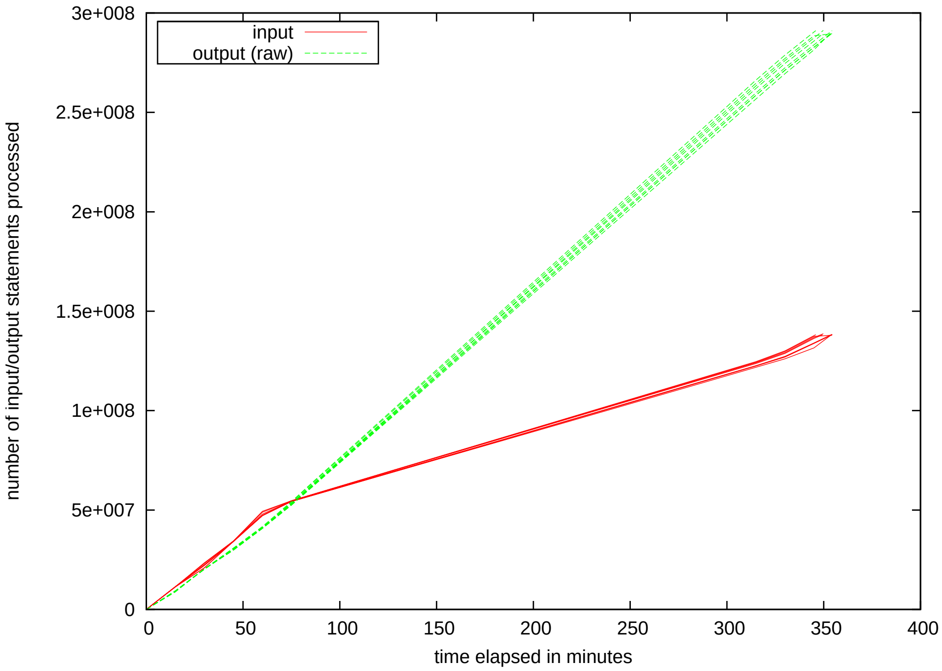


Figure 6.1: Input/output throughput during distributed assertional reasoning overlaid for each slave machine

14 min; (iv) locally grounding the T-atoms of the rules with non-empty ABody took 2 min; (v) parallel assertional reasoning took 6 h; (vi) scattering the results of the T-Box reasoning from the master machine to the slave machines took 3 min; (vii) sorting and coordinating the reasoned data by subject over the slave machines took 4.6 h; (viii) aggregating the rank annotations for all triples to produce the final optimal output (above the given threshold) took 2.7 h. Just over half of the total time is spent in the final aggregation of the optimal reasoning output. Comparing the performance of the first five steps against their classical equivalent, notably preparing the T-Box on the master machine took ∼60% longer, and assertional reasoning took ∼3× longer; we will now look more in-depth at these sub-tasks.

In total, 1.1 million (∼0.1%) T-Box triples were extracted. T-Box level reasoning produced an additional 2.579 million statements. The average rank annotation of the input T-facts was 9.94×10^{-4}, whereas the average rank annotation of the reasoned T-facts was 3.67×10^{-5}. Next, 291 thousand non-equivalent, optimal T-ground rules were produced for the assertional program, within which, 1.655 million dependency links were materialised.

In Figure 6.1, we overlay the input/output performance of each slave machine during the assertional reasoning scan—notably, the profile of each machine is very similar. During assertional reasoning, 2.232 billion raw (authoritative) inferences were created, which were immediately filtered down to 1.879 billion inferences by removing non-RDF and tautological triples—we see that 1.41×/1.68× (pre-/post-filtering) more raw inferences are created than for the classical variation. Notably, the LRU cache detected and filtered a total of 12.036 billion duplicate/dominated statements. Of the 1.879 billion inferences, 1.866 billion (99.34%) inherited their annotation from an assertional fact (as opposed to a T-ground rule), seemingly since terminological facts are generally more highly ranked by our approach than assertional facts (cf. Table 4.5).

In the final aggregation of rank annotations, from a total of 2.987 billion input/inferred statements, 1.889 billion (63.2%) unique and optimal triples were extracted; of the filtered, 1.008 billion (33.7%) were duplicates with the same annotation,[15] 89 million were (properly) dominated reasoned triples (2.9%), and 1.5 million (0.05%) were (properly) dominated asserted triples. The final average rank annotation for the aggregated triples was 5.29×10^{-7}.

6.4.3 Repair: Implementation/Evaluation

In this section, we discuss our implementation for handling inconsistencies in the annotated corpus (including asserted and reasoned data). We begin by describing our distributed approach to detect and extract sets of annotated facts which together constitute an inconsistency (i.e., a constraint violation). We then continue by discussing our approach to subsequently repair the annotated corpus.

Detecting Inconsistencies: Implementation

In Table B.6, we provide the list of OWL 2 RL/RDF constraint rules which we use to detect inconsistencies. The observant reader will note that these rules require assertional joins (have mulitple A-atoms) which we have thus far avoided in our approach. However, up to a point, we can leverage a similar algorithm to that presented in the previous section for reasoning. First, we note that the rules are by their nature not recursive (have empty heads). Second, we postulate that many of the ungrounded atoms will have a high selectivity (have a low number of ground instances in the knowledge-base). In particular, if we assume that only one atom in each constraint rule has a low selectivity, such rules are amenable to computation using the partial-indexing approach: any atoms with high selectivity are ground in-memory to create a set of (partially) grounded rules, which are subsequently flooded across all machines. Since the initial rules are not recursive, the set of (partially) grounded rules remains static. Assuming that at most one atom has low selectivity, we can efficiently apply our single-atom rules in a distributed scan as per the previous section.

However, the second assumption may not always hold: a constraint rule may contain more than one low-selectivity atom. In this case, we manually apply a distributed on-disk merge-join operation to ground the remaining atoms of such rules.

Note that during the T-Box extraction in the previous section, we additionally extract the T-atoms for the constraint rules, and apply authoritative analysis analogously (see Example 6.9).

Thus, the distributed process for extracting constraint violations is as follows:

1. **local:** apply an authoritative T-grounding of the constraint rules in Table B.6 from the T-Box resident on the master machine;

2. **flood/run:** flood the non-ground A-atoms in the T-ground constraints to all slave machines, which extract the selectivity (number of ground instances) of each pattern for their segment of the corpus, and locally buffer the instances to a separate corpus;

3. **gather:** gather and aggregate the selectivity information from the slave machines, and for each T-ground constraint, identify A-atoms with a selectivity below a given threshold (in-memory capacity);

4. ***reason:*** for rules with zero or one low-selectivity A-atoms, run the distributed reasoning process described previously, where the highly-selective A-atoms can be considered "honourary T-atoms";

5. **run(/coordinate):** for any constraints with more than one low-selectivity A-atom, apply a manual on-disk merge-join operation to complete the process.

The end result of this process is sets of annotated atoms constituting constraint violations distributed across the slave machines.

[15] Note that this would include the 171 million duplicate asserted triples from the input.

Listing 6.1: Strongest constraint violation

```
# ABox Source - http://rdf.freebase.com/rdf/type/key/namespace
# Annotation - <nb,a,0.001616>
 (fb:type.key.namespace, fb:type.property.unique, "True"^^xsd:integer)
```

Detecting Inconsistencies: Evaluation

Distributed extraction of the inconsistencies from the aggregated annotated data took, in total, 2.9 h. Of this: (i) 2.6 min were spent building the authoritative T-ground constraint rules from the local T-Box on the master machine; (ii) 26.4 min were spent extracting—in parallel on the slave machines—the cardinalities of the A-atoms of the T-ground constraint bodies from the aggregated corpus; (iii) 23.3 min were spent extracting ground instances of the high-selectivity A-atoms from the slave machines; (iv) 2 h were spent applying the partially-ground constraint rules in parallel on the slave machines.

In total, 301 thousand constraint violations were found; in Table 6.1, we give a breakdown showing the number of T-ground rules generated, the number of total ground instances (constraint violations found), and the total number of unique violations found (a constraint may fire more than once over the same data, where for example in rule cax-dw, `?c1` and `?c2` can be ground interchangeably). Notably, the table is very sparse: we highlight the constraints requiring new OWL 2 constructs in italics, where we posit that the observable lack of OWL 2 constraint axioms (and the complete lack of violations) is perhaps due to the fact that OWL 2 has not yet had enough time to gain traction on the Web (cf. Table 5.2). In fact, all of the T-ground prp-irp and prp-asyp rules come from one document[16], and all cax-adc T-ground rules come from one directory of documents[17]. Further, the only two constraint rules with violations in our Linked Data corpus were dt-not-type (97.6%) and cax-dw (2.4%).

Overall, the average violation rank degree was 1.19×10^{-7} (vs. an average rank-per-fact of 5.29×10^{-7} in the aggregated corpus). The single strongest violation degree is given in Listing 6.1, where the constraint dt-not-type—which checks for invalid datatype memberships—detects the term `"True"^^xsd:integer` as being invalid. In fact, the document serving this triple is ranked 23rd overall out of our 3.915 million sources—indeed, it seems that even highly ranked documents are prone to publishing errors and inconsistencies. Similar inconsistencies were also found with similar strengths in other documents within the FreeBase domain.[18] Thus, only a minute fraction (~0.0006%) of our corpus is above the consistency threshold.

With respect to cax-dw, we give the top 10 pairs of disjoint classes in Table 6.2, where most are related to FOAF. The single strongest violation degree for cax-dw is given in Listing 6.2, where we see that the inconsistency is given by one document, and may be attributable to use of properties without verifying their defined domain. Arguably, the entity `kingdoms:Aa` is unintentionally a member of both of the FOAF disjoint classes, where the entity is explicitly a member of `geospecies:KingdomConcept`.

Taking a slightly different example, the cax-dw violation involving the strongest assertional fact is provided in Listing 6.3, where we see a conflict between a statement asserted in thousands of documents, and a statement inferable from a single document.

Of the cax-dw constraint violations, 3,848 (54.1%) involved two assertional facts with the same annotation (such as in the former cax-dw example—likely stemming from the same assertional document). All of the constraint violations were given by assertional data (i.e., an assertional fact represented the weakest element of each violation).

[16] `http://models.okkam.org/ENS-core-vocabulary#country_of_residence`; retr. 2011/01/22

[17] `http://ontologydesignpatterns.org/cp/owl/fsdas/`; retr. 2011/01/22

[18] Between our crawl and time of writing, these errors have been fixed.

Listing 6.2: Strongest multi-triple constraint violation

```
# TBox Source - foaf: (amongst others)
# Annotations - <nb,a,0.024901>
 (foaf:primaryTopic, rdfs:domain, foaf:Document)
 (foaf:Document, owl:disjointWith, foaf:Agent)

# TBox Source - geospecies:
# Annotation - <nb,a,0.000052>
 (geospecies:hasWikipediaArticle, rdfs:domain, foaf:Person)

# ABox Source - kingdoms:Aa?format=rdf
# Annotations - <nb,a,0.000038>
 (kingdoms:Aa, foaf:primaryTopic, kingdoms:Aa)
 (kingdoms:Aa, geospecies:hasWikipediaArticle, wiki:Animal)

# Violation
# Annotations - <nb,a,0.000038>
 (kingdoms:Aa, rdf:type, foaf:Document) # Inferred
 (kingdoms:Aa, rdf:type, foaf:Person) # Inferred
```

Listing 6.3: cax-dw violation involving strongest assertional fact

```
# TBox Source - foaf: (amongst others)
# Annotations - <nb,a,0.024901>
 (foaf:knows, rdfs:domain, foaf:Person)
 (foaf:Organization, owl:disjointWith, foaf:Person)

# Comment - dav:this refers to the company OpenLink Software
# ABox Source - 5,549 documents including dav:
# Annotation - <nb,a,0.000391>
 (dav:this, rdf:type, foaf:Organization)
# ABox Source - dav: (alone)
# Annotation - <nb,a,0.000020>
 (dav:this, foaf:knows, vperson:kidehen@openlinksw.com#this)

# Violation
# Annotation - <nb,a,0.000391>
 (dav:this, rdf:type, foaf:Organization)
# Annotation - <nb,a,0.000020>
 (dav:this, rdf:type, foaf:Person) # Inferred
```

Rule	**T-ground**	**Violations**
eq-diff1	–	0
eq-diff2	–	0
eq-diff3	–	0
eq-irp	–	0
prp-irp	10	0
prp-asyp	9	0
prp-pdw	0	0
prp-adp	0	0
prp-npa1	–	0
prp-npa2	–	0
cls-nothing	–	0
cls-com	14	0
cls-maxc1	0	0
cls-maxqc1	0	0
cls-maxqc2	0	0
cax-dw	1,772	7,114
cax-adc	232	0
dt-not-type	–	294,422

Table 6.1: Number of T-ground rules, violations, and unique violations found for each OWL 2 RL/RDF constraint rule—rules involving new OWL 2 constructs are *italicised*

#	**Class 1**	**Class 2**	**Violations**
1	foaf:Agent	foaf:Document	3,842
2	foaf:Document	foaf:Person	2,918
3	sioc:Container	sioc:Item	128
4	foaf:Person	foaf:Project	100
5	ecs:Group	ecs:Individual	38
6	skos:Concept	skos:Collection	36
7	foaf:Document	foaf:Project	26
8	foaf:Organization	foaf:Person	7
9	sioc:Community	sioc:Item	3
10	ecs:Fax	ecs:Telephone	3

Table 6.2: Top 10 disjoint-class pairs

Repairing Inconsistencies: Approach

Given a (potentially large) set of constraint violations, herein we sketch an approach for repairing the corpus from which they were derived, such that the result of the repair is a consistent corpus as defined in § 6.3.7. In particular, our repair strategy is contingent on the non-constraint rules containing only one atom in the body (as is true for our assertional program).[19] In particular, we reuse notions from the seminal work of Reiter [1987] on diagnosing faulty systems.

For the moment—and unlike loosely related works on debugging unsatifiable concepts in OWL terminologies—we only consider repair of assertional data: all of our constraints involve some assertional data, and we consider terminological data as correct. Although this entails the possibility of removing atoms above the degree of a particular violation in order to repair that violation, we recall from our empirical

[19] Note that a more detailed treatment of repairing inconsistencies on the Web is currently out of scope, and would deserve a more dedicated analysis in future work. Herein, our aim is to sketch one particular approach feasible in our scenario.

analysis that 99.34% of inferred annotations are derived from an assertional fact.[20] Thus, we can reduce our repair to being with respect to the T-ground program $P = P^C \cup P^R$, where P^C is the set of proper T-ground constraint rules, and P^R is the set of proper T-ground rules in the assertional program. Again, given that each of our constraints requires assertional knowledge—i.e., that the T-ground program P only contains *proper* constraint rules—P is necessarily consistent.

Moving forward, we introduce some necessary definitions adapted from [Reiter, 1987] for our scenario. Firstly, we give:

> **The Principle of Parsimony:** *A diagnosis is a conjecture that some minimal set of components are faulty.*
>
> —**Reiter [1987]**

This captures our aim to find a non-trivial (minimal) set of assertional facts which diagnose the inconsistency of our model. Next, we define a *conflict set* which denotes a set of inconsistent facts, and give a *minimal conflict set* which denotes the least set of facts which preserves an inconsistency with respect to a given program P (note that we leave rule/fact annotations implicit in the notation):

Definition 6.13 (Conflict set) *A* conflict set *is a Herbrand interpretation $C = \{F_1, \ldots, F_n\}$ such that $P \cup C$ is inconsistent.*

Definition 6.14 (Minimal conflict set) *A* minimal conflict set *is a Herbrand interpretation $C = \{F_1, \ldots, F_n\}$ such that $P \cup C$ is inconsistent, and for every $C' \subset C$, $P \cup C'$ is consistent.*

Next, we define the notions of a hitting set and a minimal hitting set as follows:

Definition 6.15 (Hitting set) *Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ be a set of Herbrand interpretations, and $H = \{F_1, \ldots, F_n\}$ be a single Herbrand interpretation. Then, H is a* hitting set *for $\mathcal{I}$ iff for every $I_j \in \mathcal{I}$, $H \cap I_j \neq \emptyset$.*

Definition 6.16 (Minimal hitting set) *A* minimal hitting set *for $\mathcal{I}$ is a hitting set H for $\mathcal{I}$ such that for every $H' \subset H$, H' is not a hitting set for $\mathcal{I}$.*

Given a set of minimal conflict sets $\mathcal{C}$, the set of corresponding minimal hitting sets $\mathcal{H}$ represents a set of diagnoses thereof [Reiter, 1987]; selecting one such minimal hitting set and removing all of its members from each set in $\mathcal{C}$ would resolve the inconsistency for each conflict set $C \in \mathcal{C}$ [Reiter, 1987].

This leaves three open questions: (i) how to compute the minimal conflict sets for our reasoned corpus; (ii) how to compute and select an appropriate hitting set as the diagnosis of our inconsistent corpus; (iii) how to repair our corpus with respect to the selected diagnosis.

Computing the (extended) minimal conflict sets

In order to compute the set of minimal conflict sets, *we leverage the fact that the program P^R does not contain rules with multiple A-atoms in the body.*

First, we must consider the fact that our corpus Γ already represents the least model of $\Gamma \cup P^R$ and thus define an *extended minimal conflict set* as follows:

[20] Note also that since our T-Box is also part of our A-Box, we may defeat facts which are terminological, but only based on inferences possible through their assertional interpretation.

Definition 6.17 (Extended minimal conflict set) *Let Γ be a Herbrand model such that $\Gamma = \mathsf{lm}(\Gamma \cup P^R)$, and let $C = \{F_1, \ldots, F_n\}, C \subseteq \Gamma$ denote a minimal conflict set for Γ. Let*

$$\mathsf{extend}(F) = \{F' \in \Gamma \mid F \in \mathsf{lm}(P^R \cup \{F'\})\}$$

be the set of all facts in Γ from which some F can be derived wrt. the linear program P^R (clearly $F \in \mathsf{extend}(F)$). We define the extended minimal conflict set (EMCS) for C wrt. Γ and P^R as a collection of sets $\mathcal{E} = \{\mathsf{extend}(F) \mid F \in C\}$.

Thus, given a minimal conflict set, the extended minimal conflict set encodes choices of sets of facts that must be removed from the corpus Γ to repair the violation, such that the original *seed fact* cannot subsequently be re-derived by running the program P^R over the reduced corpus. The concept of a (minimal) hitting set for a collection of EMCSs follows naturally and similarly represents a diagnosis for the corpus Γ.

To derive the complete collection of ECMSs from our corpus, we sketch the following process. Firstly, for each constraint violation detected, we create and load an initial (minimal) conflict set into memory; from this, we create an extended version representing each member of the original conflict set (*seed fact*) by a singleton in the extended set. (Internally, we use a map structure to map from facts to the extended set(s) that contain it, or or course, **null** if no such conflict set exists). We then reapply P^R over the corpus in parallel, such that—here using notation which corresponds to Algorithm 5.1—for each input triple t being reasoned over, for each member t_δ of the subsequently inferred set G_n, if t_δ is a member of an EMCS, we add t to that EMCS.

Consequently, we populate the collection of EMCS, where removing all of the facts in one member of each EMCS constitutes a repair (a diagnosis). With respect to distributed computation of the EMCSs, we can run the procedure in parallel on the slave machines, and subsequently merge the results on the master machine to derive the global collection of ECMSs for subsequent diagnosis.[21]

Preferential strategies for annotated diagnoses

Before we continue, we discuss two competing models for deciding an appropriate diagnosis for subsequent reparation of the annotated corpus. Consider a set of violations that could be solved by means of removing one 'strong' fact—e.g., a single fact associated with a highly-ranked document—or removing many weak facts—e.g., a set of facts derived from a number of low-ranked documents: should one remove the strong fact or the set of weak facts? Given that the answer is non-trivial, we identify two particular means of deciding a suitable diagnosis: i.e., we identify the characteristics of an appropriate minimal hitting set with respect to our annotations. Given any such quantitative strategies, selecting the most appropriate diagnosis then becomes an optimisation problem.

Strategy 1: we prefer a diagnosis which minimises the number of facts to be removed in the repair. This can be applied independently of the annotation framework. However, this diagnosis strategy will often lead to trivial decisions between elements of a minimal conflicting set with the same cardinality; also, we deem this strategy to be vulnerable to spamming such that a malicious low-ranked document may publish a number of facts which conflict and defeat a fact in a high-ranked document. Besides spamming, in our repair process, it may also trivially favour, e.g., memberships of classes which are part of a deep class hierarchy (the memberships of the super-classes would also need to be removed).

Strategy 2: we prefer a diagnosis which minimises the strongest annotation to be removed in the repair. This has the benefit of exploiting the granular information in the annotations, and being computable with

[21] Note in fact that instead of maintaining a set of ECMSs, to ensure a correct merge of ECMSs gathered from the slave machines, we require an ordered sequence of conflict sets.

the glb/lub functions defined in our annotation framework; however, for general annotations in the domain $\mathbf{D}$ only a *partial-ordering* is defined, and so there may not exist an unambiguous strongest/weakest annotation—in our case, with our predefined threshold removing blacklisted and non-authoritative inferences from the corpus, we need only consider rank annotations for which a total-ordering is present. Also, this diagnosis strategy may often lead to trivial decisions between elements of a minimal conflicting set with identical annotations—in our case, most likely facts from the same document which we have seen to be a common occurrence in our constraint violations (54.1% of the total *raw* cax-dw violations we empirically observe).

Strategy 3: we prefer a diagnosis which minimises the total sum of the rank annotation involved in the diagnosis. This, of course, is domain-specific and also falls outside of the general annotation framework, but will likely lead to less trivial decisions between equally 'strong' diagnoses. In the naïve case, this strategy is also vulnerable to spamming techniques, where one 'weak' document can make a large set of weak assertions which culminate to defeat a 'strong' fact in a more trustworthy source.

In practice, we favour *Strategy 2* as exploiting the additional information of the annotations and being less vulnerable to spamming; when *Strategy 2* is inconclusive, we resort to *Strategy 3* as a more granular method of preference, and thereafter if necessary to *Strategy 1*. If all preference orderings are inconclusive, we then select an arbitrary syntactic ordering.

Going forward, we formalise a total ordering $\leq_I$ over a pair of (annotated) Herbrand interpretations which denotes some ordering of preference of diagnoses based on the 'strength' of a set of facts—a stronger set of facts (alternatively, a set which is less preferable to be removed) denotes a higher order. The particular instantiation of this ordering depends on the repair strategy chosen, which may in turn depend on the specific domain of annotation.

Towards giving our notion of $\leq_I$, let I_1 and I_2 be two Herbrand interpretations with annotations from the domain $\mathbf{D}$, and let $\leq_{\mathbf{D}}$ denote the partial-ordering defined for $\mathbf{D}$. Starting with *Strategy 2*—slightly abusing notation—if $\mathsf{lub}(\{I_1\}) <_{\mathbf{D}} \mathsf{lub}(\{I_1\})$, then $I_1 <_r I_2$; if $\mathsf{lub}(\{I_1\}) >_{\mathbf{D}} \mathsf{lub}(\{I_2\})$, then $I_1 >_I I_2$; otherwise (if $\mathsf{lub}(\{I_1\}) =_{\mathbf{D}} \mathsf{lub}(\{I_2\})$ or $\leq_{\mathbf{D}}$ is undefined for I_1,I_2), we resort to *Strategy 3* to order I_1 and I_2: we apply a domain-specific "summation" of annotations (ranks) denoted $\Sigma_{\mathbf{D}}$ and define the order of I_1, I_2 such that if $\Sigma_{\mathbf{D}}\{I_1\} <_{\mathbf{D}} \Sigma_{\mathbf{D}}\{I_2\}$, then $I_1 <_I I_2$, and so forth. If still equals (or uncomparable), we use the cardinality of the sets, and thereafter consider an arbitrary syntactic order. Thus, sets are given in ascending order of their single strongest fact (*Strategy 2*), followed by the order of their rank summation (*Strategy 3*), followed by their cardinality (*Strategy 1*), followed by an arbitrary syntactic ordering. Note that $I_1 =_I I_2$ iff $I_1 = I_2$. Given $\leq_I$, the functions max_I and min_I follow naturally.

Computing and selecting an appropriate diagnosis

Given that we have $\sim 7 \times 10^3$ non-trivial (extended) conflict sets—i.e., conflict sets with cardinality greater than one—we would wisely wish to avoid materialising all $2^{7\times10^3}$ hitting sets. Similarly, we wish to avoid expensive optimisation techniques [Stuckenschmidt, 2008] for deriving the minimal diagnosis with respect to $\leq_I$. Instead, we use a heuristic to materialise one hitting set which gives us an *appropriate*, but possibly sub-optimal diagnosis. Our diagnosis is again a flat set of facts, which we denote by Δ.

First, to our diagnosis we immediately add the union of all members of singleton (trivial) EMCSs, where these are necessarily part of any diagnosis. This would include, for example, all facts which must be removed from the corpus to ensure that no violation of dt-not-type can remain or be re-derived in the corpus.

For the non-trivial EMCSs, we first define an ordering of conflict sets based on the annotations of its members, and then cumulatively derive a diagnosis by means of a descending iteration of the ordered sets For the ordered iteration of the ECMS collection, we must define a total ordering $\leq_{\mathcal{E}}$ over $\mathcal{E}$ which directly corresponds to $\mathsf{min}_I(E_1) \leq_I \mathsf{min}_I(E_2)$—a comparison of the weakest set in both.

We can then apply the following diagnosis strategy: iterate over $\mathcal{E}$ in descending order wrt. $\leq_{\mathcal{E}}$, such that

$$\forall E \in \mathcal{E},\ \text{if } \nexists I \in E (I \subseteq \Delta) \text{ then } \Delta = \Delta \cup \mathsf{min}_I(E)$$

where after completing the iteration, the resulting Δ represents our diagnosis. Note of course that Δ may not be optimal according to our strategy, but we leave further optimisation techniques for a later scope.

Repairing the corpus

Removing the diagnosis Δ from the corpus Γ will lead to consistency in $P \cup (\Gamma \setminus \Delta)$. However, we also wish to remove the facts that are inferable through Δ with respect to P, which we denote as Δ^+. We also want to identify facts in Δ^+ which have alternative derivations from the non-diagnosed input data ($\Gamma_{raw} \setminus \Delta$), and include them in the repaired output, possibly with a weaker annotation: we denote this set of re-derived facts as Δ^-. Again, we sketch an approach contingent on P only containing proper rules with one atom in the body.

First, we determine the set of statements inferable from the diagnosis, given as:

$$\Delta^+ = \mathsf{lm}(P \cup \Delta) \setminus \Delta\,.$$

Secondly, we scan the *raw* input corpus Γ_{raw} as follows. First, let $\Delta^- := \{\}$. Let

$$\Gamma_{raw}^{\cancel{\Delta}} := \{F{:}\mathbf{d} \in \Gamma_{raw} \mid \nexists \mathbf{d}'(F{:}\mathbf{d}' \in \Delta)\}$$

denote the set of annotated facts in the raw input corpus not appearing in the diagnosis. Then, scanning the raw input (ranked) data, for each $F_i \in \Gamma_{raw}^{\cancel{\Delta}}$, let

$$\delta_i^- := \{F'{:}\mathbf{d}' \in \mathsf{lm}(P \cup \{F_i{:}\mathbf{d}_i\}) \mid \exists \mathbf{d}_x(F'{:}\mathbf{d}_x \in \Delta^+), \nexists \mathbf{d}_y \geq \mathbf{d}'(F'{:}\mathbf{d}_y \in \Delta^-)\}$$

denote the intersection of facts derivable from both F_i and Δ^+ which are not dominated by a previous rederivation; we apply $\Delta_i^- := \mathsf{max}(\Delta_{i-1}^- \cup \delta_i^-)$, maintaining the dominant set of rederivations. After scanning all raw input facts, the final result is Δ^-:

$$\Delta^- = \mathsf{max}(\{F{:}\mathbf{d} \in \mathsf{lm}(P \cup \Gamma_{raw}^{\cancel{\Delta}}) \mid \exists \mathbf{d}'(F{:}\mathbf{d}' \in \Delta^+)\})$$

the dominant rederivations of facts in Δ^+ from the non-diagnosed facts of the input corpus.

Finally, we scan the entire corpus Γ and buffer any facts not in $\Delta \cup \Delta^+ \setminus \Delta^-$ to the final output, and if necessary, weaken the annotations of facts to align with Δ^-.[22]

Distributed implementation

We briefly describe the distributed implementation as follows:

- **gather**: the set of conflict sets (constraint violations) detected in the previous stages of the process are gathered onto the master machine;
- **flood/run**: the slave machines receive the conflict sets from the master machine and reapply the (positive) T-ground program over the *entire* corpus; any triple involved in the inference of a member

[22] One may again note that if there are terminological facts in Δ, the T-Box inferences possible through these facts may remain in the final corpus, even though the corpus is consistent; if required, removal of all such T-Box inferences would be possible by rerunning the entire reasoning process over $\Gamma_{raw} \setminus \Delta$—the repaired raw corpus.

of a conflict set is added to an extended conflict set;

- **gather**: the respective extended conflict sets are merged on the master machine, and the sets are ordered by $\leq_{\mathcal{E}}$ and iterated over—the initial diagnosis Δ is thus generated; the master machine applies reasoning over Δ to derive Δ^+ and **flood**s this set to the slave machines;
- **flood/run**: the slave machines rerun the reasoning over the *input* corpus to try to find alternate (non-diagnosed) derivations for facts in Δ^+ (which are added to Δ^-);
- **gather**: the set of alternate derivations are gathered and aggregated on the master machine, which prepares the final Δ^- set (maintaining only dominant rederivations in the merge);
- **flood/run**: the slave machines accept the final diagnosis and scan the *entire* corpus again, buffering a repaired (consistent) corpus.

Repair Evaluation

The total time taken for the distributed diagnosis and repair of the corpus was 2.82 h; the bulk of the time was taken for (i) extracting the extended conflict sets from the input/inferred corpus on the slave machines which took 24.5 min; (ii) deriving the alternate derivations Δ^- over the input corpus which took 18.8 min; (iii) repairing the corpus which took 1.94 h.[23]

The initial diagnosis over the extended conflict set contained 316,884 entries, and included 16,733 triples added in the extension of the conflict set (triples which inferred a member of the original conflict set). 413,744 facts were inferable for this initial diagnosis, but alternate derivations were found for all but 101,018 (24.4%) of these; additionally, 123 weaker derivations were found for triples in the initial diagnosis. Thus, the entire repair involved removing 417,912 facts and weakening 123 annotations, touching upon 0.2% of the closed corpus.

6.5 Related Work

In this section, we introduce related works in the area of (i) the field of annotated programs and annotated reasoning (§ 6.5.1); and (ii) knowledge-base repair (§ 6.5.2).

6.5.1 Annotated Reasoning

Bistarelli et al. [2008] extend the Datalog language with weights that can represent metainformation relevant to Trust Management systems, such as costs, preferences and trust levels associated to policies or credentials. Weights are taken from a *c*-semiring where the two operations $\times$ and $+$ are used, respectively, to compose the weights associated to statements, and select the best derivation chain. Some example of *c*-semirings are provided where weights assume the meaning of costs, probabilities and fuzzy values. In all of these examples, the operator $\times$ is idempotent, so that the *c*-semiring induces a complete lattice where $+$ is the lub and $\times$ is the glb. In such cases, we can use $\mathsf{opt}(P)$ to support their proposed framework using our annotated programs. The complexity of Weighted Datalog is just sketched—only decidability is proven. Scalability issues are not tackled and no experimental results are provided.

Flouris et al. [2009] tackle the provenance of inferred RDF data by augmenting triples with a fourth component named *color*, representing the collection of the different data sources used to derive a triple.

[23] Note that for the first two steps, we use an optimisation technique to skip reasoning over triples whose terms do not appear in Δ and Δ^+ respectively.

A binary operation + over colours forms a semigroup; the provenance of derived triples is the sum of the provenance of the supporting triples. This framework can be simulated in our annotated programs by adopting a flat lattice, where all the elements (representing different provenances) are mutually incomparable. Then, for each derived atom, $\mathsf{plain}(P)$ collects all the provenances employed by Flouris et al. [2009]. Although the complexity analysis yields an almost linear upper bound, no experimental results are provided.

Straccia [2010] present "SoftFacts": a top-k retrieval engine which uses an ontological layer to offer ranked results over a collection of databases—results may also be sourced from inferred knowledge. The database stores annotated facts in the form of n-ary relations with an associated score in the range $[0, 1]$; the ontology layer supports crisp axioms relating to class subsumption and intersection; then, an abstraction layer relates concepts and relations to physical storage (database tables). The system handles conjunctive queries and allows for various aggregation functions over ranking scores; top-k processing is optimised (in the absence of aggregation functions) using their Disjunctive Thresholding Algorithm (DTA). However, their experiments are limited to an ontology containing 5,115 axioms and 2,550, and data in the order of a hundred-thousand relations.

Lopes et al. [2010a] present a general annotation framework for RDFS, together with $AnQL$: a query language inspired by SPARQL which includes querying over annotations. Annotations are formalised in terms of residuated bounded lattices, which can be specialised to represent different types of meta-information (temporal constraints, fuzzy values, provenance etc.). A general deductive system—based on abstract algebraic structures—has been provided and proven to be offer PTIME complexity. The Annotated RDFS framework allows for representing a large spectrum of different meta-information. However, the framework of Lopes et al. [2010a] is strictly anchored to RDFS, while our annotated programs are founded on OWL 2 RL/RDF and hence are transversal with respect to the underlying ontology language. Moreover, our results place more emphasis on scalability.

6.5.2 Inconsistency Repair

Most legacy works in this area (e.g., see DION and MUPSTER [Schlobach et al., 2007] and a repair tool for unsatisifable concepts in Swoop [Kalyanpur et al., 2006]) focus on debugging singular OWL ontologies within a Description Logics formalism, in particular focussing on fixing terminologies (T-Boxes) which include unsatisfiable concepts—not of themselves an inconsistency, but usually indicative of a modelling error (termed incoherence) in the ontology. Such approaches usually rely on the extraction and analysis of MUPs (Minimal Unsatisfiability Preserving Sub-terminologies) and MIPs (Minimal Incoherence Preserving Sub-terminologies), usually to give feedback to the ontology editor during the modelling process. However, these approaches again focus on debugging terminologies, and have been shown in theory and in practice to be expensive to compute—please see [Stuckenschmidt, 2008] for a survey (and indeed critique) of such approaches.

Ferrara et al. [2008] look at resolving inconsistencies brought about by possibly imprecise Description Logics ontology mappings—in particular, fuzzy values are used to denote a degree of "confidence" in each mapping. Mappings are then analysed in descending order of fuzzy confidence values, and are tested with respect to an A-Box to see if they cause inconsistency; if so, the conditions of the inconsistency are used to lower the mapping confidence by a certain degree. Further, they discuss conflict resolution, where two or more competing mappings together cause inconsistency; they note that deciding which mappings to preserve is non-trivial, but choose to adopt an approach which removes the (possibly strong) mappings causing the most inconsistencies. Again, they seemingly focus on debugging and refining mappings between a small number of (possibly complex and large) ontologies, and do not present any empirical evaluation.

6.6 Critical Discussion and Future Directions

In this chapter, we have given a comprehensive discourse on using annotations to track indicators of provenance and trust during Linked Data reasoning. In particular, we track three dimensions of trust-/provenance-related annotations for data: viz. (i) *blacklisting*; (ii) *authority*; and (iii) *ranking*. We presented a formal annotated reasoning framework for tracking these dimensions of trust and provenance during the reasoning procedure. We gave various formal properties of the program—some specific to our domain of annotation, some not—which demonstrated desirable properties relating to termination, growth of the program, and efficient implementation. Later, we provided a use-case for our annotations involving detection and repair of inconsistencies. We presented implementation of our methods over a cluster of commodity hardware and evaluated our techniques with respect to our large-scale Linked Data corpus. As such, we have looked at non-trivial reasoning procedure which incorporates Linked Data principles, links-based analysis, annotated logic programs, a subset of OWL 2 RL/RDF rules (including the oft overlooked constraint rules), and inconsistency repair techniques, into a coherent system for scalable, distributed Linked Data reasoning.

However, we identify a number of current shortcomings in our approach which hint at possible future directions.

Firstly, again we do not support rules with multiple A-atoms in the body—adding support for such rules would imply a significant revision of our reasoning and inconsistency repair strategies.

Further, one may consider a much simpler approach for deriving ranked inferences: we could pre-filter blacklisted data from our corpus, apply authoritative reasoning as per the previous chapter—and since our assertional inferences can only be the result of one fact originating from one document—we can assign each such inference the context of that document, and propagate ranks through to the inferred data accordingly; however, we see the annotation framework as offering a more generic and extensible basis for tracking various metainformation during reasoning, offering insights into the general conditions under which this can be done in a scalable manner.

Along similar lines, by only considering the strongest evidence for a given derivation, we overlook the (potentially substantial) cumulative evidence given by many weaker sources: although the ranking of our input data reflects cumulative evidence, our annotated reasoning does not. However, looking at supporting a cumulative aggregation of annotations raises a number of non-trivial questions with respect to scalability and termination. Also, one may have to consider whether pieces of evidence are truly independent so as to avoid considering the cumulative effect of different "expressions" of the same evidence—as per our input ranking, this could perhaps be done based on the source of information.

With respect to Linked Data, we note that our inconsistency analysis may only be able to perceive a small amount of the noise present in the input and inferred data. Informally, we believe that Linked Data vocabularies are not sufficiently concerned with axiomatising common-sense constraints,[24] particularly those which are *clear* indicators of noise and are useful for detecting when unintended reasoning occurs—thus, more granular (possibly domain specific or heuristic) means of diagnosing problems may be required.

Finally, given a sufficient means of identifying errors in the data, it would be interesting to investigate alternative scalable repair strategies which maximise the *utility* of the resulting corpus to a consumer; however, objectively evaluating the desirability of repairs is very much an open question, which may only become answerable as Linked Data consumer applications (and their requirements) mature.

As the Web of Data expands and diversifies, we believe that the need for reasoning will grow more and more apparent, as will the implied need for methods of handling and incorporating notions of trust and

[24] For example, since the time of our crawl, we notice that maintainers of the FOAF vocabulary have removed disjointness constraints between `foaf:Person`/`foaf:Agent` and `foaf:Document`, citing possible examples where an individual could exist in both classes. In our opinion, these axioms are (were) clear indicators of noise, which give automated reasoning processes useful cues for problematic data to repair.

provenance which scale to large corpora, and which are tolerant to spamming and other malicious activity. Although there is still much work to do, we feel that the research presented in this chapter offers significant insights into how the Linked Data reasoning systems can profit from the more established area of General Annotated Programs in order to handle issues relating to data quality and provenance in a scalable, domain-independent, extensible and well-defined manner.

Chapter 7

Consolidation[*]

> *"You can know the name of a bird in all the languages of the world, but when you're finished, you'll know absolutely nothing whatever about the bird... So let's look at the bird and see what it's doing – that's what counts. I learned very early the difference between knowing the name of something and knowing something."*
>
> —**Richard Feynman**

Thus far—and with the exception of (non-recursive) constraints—we have looked at applying reasoning exclusively over rules which do not require assertional joins, citing computational expense and the potential for quadratic or cubic growth in materialisation as reasons not to support a fuller profile of OWL 2 RL/RDF rules. Along these lines, many of the optimised algorithms presented in the previous two chapters rely on the supported rules containing, at most, one A-atom. However, almost all OWL 2 RL/RDF rules which support the semantics of equality for resources (individuals) require the computation of assertional joins: these include rules which can be used to ascertain equality and produce `owl:sameAs` relations (listed in Table B.8; viz., prp-fp, prp-ifp, prp-key, cax-maxc2, cls-maxqc3, cls-maxqc4) and rules which axiomatise the consequences of equality, including the transitivity, symmetry and reflexivity of the `owl:sameAs` relation, as well as the semantics of replacement (listed in Table B.8; viz., eq-ref, eq-sym, eq-trans, eq-rep-s, eq-rep-p, eq-rep-o).

As discussed in the introduction of this thesis (in particular, § 1.1.1), we expect a corpus, such as ours, collected from millions of Web sources to feature significant *coreference*—use of different identifiers to signify the same entity—where the knowledge contribution on that entity is fractured by the disparity in naming. Consumers of such a corpus may struggle to achieve complete answer for their queries; again, consider a simple example query:

What are the webpages related to Tim Berners-Lee?

Knowing that Tim uses the URI `timblfoaf:i` to refer to himself in his personal FOAF profile document, and again knowing that the property `foaf:page` defines the relationship from resources to the documents somehow concerning them, we can formulate the SPARQL query given in Listing 7.1.

However, other publishers use different URIs to identify Tim, where to get more complete answers across these naming schemes, the SPARQL query must (as per the example at the outset of Chapter 5) use disjunctive `UNION` clauses for each known URI; we give an example in Listing 7.2 using identifiers from our Linked Data corpus.

[*] Parts of this chapter have been published as [Hogan et al., 2009b, 2010d] with newer results submitted for review as [Hogan et al., 2010b,e].

Listing 7.1: Simple query for pages relating to Tim Berners-Lee

```
SELECT ?page
WHERE {
  timblfoaf:i foaf:page ?page .
}
```

Listing 7.2: Extended query for pages relating to Tim Berners-Lee (sic.)

```
SELECT ?page
WHERE {
  UNION { timblfoaf:i foaf:page ?page . }
  UNION { identicauser:45563 foaf:page ?page . }
  UNION { dbpedia:Berners-Lee foaf:page ?page . }
  UNION { dbpedia:Dr._Tim_Berners-Lee foaf:page ?page . }
  UNION { dbpedia:Dr._Tim_Berners_Lee foaf:page ?page . }
  UNION { dbpedia:Sir_Timothy_John_Berners-Lee foaf:page ?page . }
  UNION { dbpedia:Tim-Berners_Lee foaf:page ?page . }
  UNION { dbpedia:TimBL foaf:page ?page . }
  UNION { dbpedia:Tim_Berners-Lee foaf:page ?page . }
  UNION { dbpedia:Tim_Bernes-Lee foaf:page ?page . }
  UNION { dbpedia:Tim_Bernes_Lee foaf:page ?page . }
  UNION { dbpedia:Tim_Burners_Lee foaf:page ?page . }
  UNION { dbpedia:Tim_berners-lee foaf:page ?page . }
  UNION { dbpedia:Timbl foaf:page ?page . }
  UNION { dbpedia:Timothy_Berners-Lee foaf:page ?page . }
  UNION { dbpedia:Timothy_John_Berners-Lee foaf:page ?page . }
  UNION { yago:Tim_Berners-Lee foaf:page ?page . }
  UNION { fb:en.tim_berners-lee foaf:page ?page . }
  UNION { fb:guid.9202a8c04000641f80000000003b0a foaf:page ?page . }
  UNION { swid:Tim_Berners-Lee foaf:page ?page . }
  UNION { dblp:100007 foaf:page ?page . }
  UNION { avtimbl:me foaf:page ?page . }
  UNION { bmpersons:Tim+Berners-Lee foaf:page ?page . }
  ...
}
```

In this example, we use (a subset of) real coreferent identifiers for Tim Berners-Lee taken from Linked Data, where we see disparate URIs not only across data publishers, but also within the same namespace. Thus (again), the expanded query quickly becomes extremely cumbersome.[1]

In this chapter, we look at bespoke methods for identifying and processing coreference in a manner such that the resultant corpus can be consumed as if more complete agreement on URIs was present; in other words, using standard query-answering techniques, we want the enhanced corpus to return the same answers for the original simple query as for the latter expanded query.

Towards this goal, we identify three high-level steps:[2]

[1] Combined with the terminological permuations for `foaf:page` exemplified at the outset of Chapter 5—and considering additional hetereogeneous query patterns such as `rdfs:label`—this query quickly demonstrates the difficulty of achieving a comprehensive set of answers over the raw corpus without reasoning. Also, such examples suggest that naïve query rewriting methods may struggle for large, heterogeneous Linked Data corpora such as ours.

[2] We note that in theory, disambiguation should come before canonicalisation—one finalises the coreference information before applying canonicalisation so as to avoid having to later revert parts of this process—however, in practice, our current disambiguation techniques can only be applied over canonicalised data.

1. **determine coreference** between identifiers (i.e., equivalence between resources);
2. **canonicalise** coreferent identifiers in the corpus;
3. **disambiguate** by identifying and repairing problematic coreference (e.g., as caused by noisy data), subsequently reverting canonicalisation where appropriate.

For determining coreference, we will rely on (i) explicit `owl:sameAs` information provided by publishers, and (ii) `owl:sameAs` information additionally inferable through OWL 2 RL/RDF rules; thus, our coreference is correct (but incomplete) with respect to the semantics of the data. However, given the nature of our corpus, we realise that some subset of this coreference information may be attributable to noise, naïve publishing, etc.; thus we include (iii) a disambiguation step to try and pinpoint potentially unintended coreference.

Along these lines, we identify the following requirements for this task in our given scenario:

- the component *must* give **high precision** of consolidated results;
- the underlying algorithm(s) *must* be **scalable**;
- the approach *must* be **fully automatic**;
- the methods *must* be **domain agnostic**;

where a component with poor precision will lead to a garbled canonicalised corpus merging disparate entities, where scalability is required to apply the process over our corpora typically in the order of a billion statements, where the scale of the corpora under analysis precludes any manual intervention, and where—for the purposes of the presented thesis at least—the methods should not give preferential treatment to any domain or vocabulary of data (other than core RDF(S)/OWL terms). Alongside these *primary requirements*, we also identify the following *secondary criteria*:

- the analysis *should* demonstrate **high recall**;
- the underlying algorithm(s) *should* be **efficient**;

where the consolidation component should identify as many (correct) equivalences as possible, and where the algorithm should be applicable in reasonable time. Clearly the secondary requirements are also important, but they are superceded by those given earlier, where a certain trade-off exists: we prefer a system which gives a high percentage of correct results and leads to a clean consolidated corpus over an approach which gives a higher percentage of consolidated results but leads to a partially garbled corpus; similarly, we prefer a system which can handle more data (is more scalable), but may possibly have a lower throughput (is less efficient).

Along these lines, in this chapter we look at methods for scalable, precise, automatic and domain-agnostic entity consolidation over large, static Linked Data corpora. Following the precedent (and rationale) laid out in previous chapters, in order to make our methods *scalable*, we avoid dynamic on-disk index structures and instead opt for algorithms which rely on sequential on-disk reads/writes of compressed flat files, again using operations such as scans, external sorts, merge-joins, and only light-weight or non-critical in-memory indices. In order to make our methods *efficient*, we again demonstrate distributed implementations of our methods over a cluster of shared-nothing commodity hardware, where our algorithms attempt to maximise the portion of time spent in embarrassingly parallel execution—i.e., parallel, independent computation without need for inter-machine coordination. In order to make our methods *domain-agnostic* and *fully-automatic*, we exploit the generic formal semantics of the data described in RDF(S)/OWL and also, generic statistics derivable from the corpus. In order to achieve *high recall*, we attempt to exploit—insofar as possible—both the formal semantics and the statistics derivable from the corpus to identify equivalent entities. Aiming at *high precision*, we introduce methods which again exploit the semantics and statistics of the data, but to conversely disambiguate entities—to defeat equivalences found in the previous step which are unlikely to be true according to some criteria.

In particular, in this chapter, we:

- discuss aspects of the OWL semantics relating to equality (§ 7.1);
- characterise our evaluation corpus with respect to the (re)use of identifiers across sources (§ 7.2);
- describe and evaluate our distributed base-line approach for consolidation which leverages explicit `owl:sameAs` relations (§ 7.3);
- describe and evaluate a distributed approach which extends consolidation to consider richer features of the OWL semantics useful for consolidation (§ 7.4);
- present a distributed algorithm for determining a form of weighted similarity—which we call *concurrence*—between entities using statistical analysis of predicates in the corpus (§ 7.5/§ D);
- present a distributed approach to disambiguate entities—i.e., detect coreference which is likely to be erroneous—combining the semantics and statistics derivable from the corpus (§ 7.6);
- render related work (§ 7.7);
- conclude the chapter with discussion (§ 7.8).

Note that we wish to decouple reasoning and consolidation, where a consumer may require using *either* or *both*, depending on the scenario; thus, in this chapter, although we incorporate reasoning techniques from Chapter 5, *we assume that consolidation is to be applied directly over the input corpus as produced by the crawler.* The same methods can be analogously applied over the merge of the (annotated) input and inferred data (as required).

7.1 OWL Equality Semantics

OWL 2 RL/RDF rules [Grau et al., 2009] support a partial-axiomatisation of the OWL 2 RDF-Based Semantics of equality, where equality between two resources is denoted by an `owl:sameAs` relation.

Firstly, in Table B.8, we provide the rules which use terminological knowledge (alongside assertional knowledge) to directly infer `owl:sameAs` relations (viz., prp-fp, prp-ifp, *prp-key*, cax-maxc2, *cls-maxqc3*, *cls-maxqc4*); we identify rules which require new OWL 2 constructs by italicising the rule label. We note that applying these rules in isolation may not be sufficient to derive a complete set of `owl:sameAs` relations: the bodies of such rules may be (partially) instantiated by inferred facts, such that data inferred through the reasoning techniques described in the previous two chapters may (typically indirectly) lead to the derivation of new `owl:sameAs` data. This will be further discussed in § 7.4.

In this chapter, we will again use the constraint rules for finding inconsistencies (enumerated in Table B.6), in this case to detect possible incorrect coreference; for example, OWL allows for asserting *in*equality between resources through the `owl:differentFrom` relation, which can be used to disavow coreference: when `owl:sameAs` and `owl:differentFrom` coincide, we err on the side of caution, favouring the latter relation and revising the pertinent coreference. Similarly, when coreference is found to be the cause of *novel* inconsistency—what we call *unsatisfiable coreference*—we cautiously repair the equality relationships involved. This process will be discussed further in § 7.6.

In Table B.7, we provide the set of rules which support the (positive) semantics of `owl:sameAs`, axiomatising the reflexivity (eq-ref), symmetry (eq-sym) and transitivity (eq-trans) of the relation, as well as support for the semantics of replacement (eq-rep-*). Note that we (optionally, and in the case of later evaluation) do not support eq-ref or eq-rep-p, and provide only partial support for eq-rep-o: (i) although eq-ref will lead to a large bulk of materialised reflexive `owl:sameAs` statements, it is not difficult to see that such statements will not lead to any consolidation or other non-reflexive equality relations; (ii) given that we operate over unverified Web data—and indeed that there is much noise present in such data—we do not want possibly imprecise equality relations to affect predicates of triples, where we support inferencing on such "terminological positions" using the reasoning techniques of the previous chapter (we assume that the more specific `owl:equivalentProperty` relation is used to denote equality between properties); (iii) for similar reasons, we do not support replacement for terms in the object position of `rdf:type` triples (we assume that the more specific `owl:equivalentClass` relation is used to denote equality between classes, in line with the spirit of punning [Golbreich and Wallace, 2009]). Finally, herein we do not consider consolidation of literals; one may consider useful applications, such as for canonicalising datatype literals (e.g., canonicalising literals such as `"1.0"^^xsd:decimal` and `"1"^^xsd:integer` which have the same data value, as per OWL 2 RL/RDF rule dt-eq), but such discussion is out of the current scope where we instead focus on finding coreference between (skolem) blank-node and URI identifiers which (in our scenario) are directly referent to entities.

Given that the semantics of equality is quadratic with respect to the assertional data, we apply a partial-materialisation approach which gives our notion of *consolidation*—instead of materialising all possible inferences given by the semantics of replacement, we instead choose one canonical identifier to represent the set of equivalent terms. We have used this approach in previous works [Hogan et al., 2007a, 2009b, 2010b], and it has also appeared in related works in the literature [Kiryakov et al., 2009; Urbani et al., 2010; Kolovski et al., 2010; Bishop et al., 2011], as a common-sense optimisation for handling data-level equality. To take an example, in previous works [Hogan et al., 2007a] we found a valid *equivalence class* (a set of coreferent identifiers) with 32,390 members; materialising all non-reflexive `owl:sameAs` statements would infer more than 1 billion `owl:sameAs` relations ($32{,}390^2$ - 32,390 = 1,049,079,710); further assuming that each entity appeared in, on average, two quadruples, we would infer an additional ~2 billion statements of massively duplicated data.

Note that although we only perform partial materialisation—and with the exception of not supporting eq-ref-p and only partially supporting eq-rep-o—we do not change the semantics of equality: alongside the partially materialised data, we provide a set of consolidated `owl:sameAs` relations (containing all of the identifiers in each equivalence class) which can be used to "backward-chain" the full inferences possible through replacement (as required).[3] Thus, we do not consider the chosen canonical identifier as somehow 'definitive' or superceding the other identifiers, but merely consider it as *representing* the equivalence class.

7.2 Corpus: Naming Across Sources

We now briefly characterise our corpus with respect to the usage of identifiers across data sources, thus rendering a picture of the morphology of the data which are subject to our consolidation approach. Again, since we do not consider the consolidation of literals or schema-level concepts, we focus on surveying the use of terms in a *data-level position*: viz., non-literal terms in the subject position, or in the object position of non-`rdf:type` triples, which typically denote individuals (as opposed to terms in the predicate position or object position of `rdf:type` triples which refer to properties and classes respectively—instead, please see

[3] With respect to eq-ref, one can consider fairly trivial backward-chaining (or query-time) support for said semantics.

§ 4.2.2 for statistics on these latter two categories of terms).[4]

We found 286.3 million unique terms, of which 165.4 million (57.8%) were blank-nodes, 92.1 million (32.2%) were URIs, and 28.9 million (10%) were literals. With respect to literals, each had on average 9.473 data-level occurrences (by definition, all in the object position).

With respect to blank-nodes—which, by definition, cannot be reused across documents (§ 3.1)—each had on average 5.233 data-level occurrences. Each occurred on average 0.995 times in the object position of a non-`rdf:type` triple, with 3.1 million (1.87%) not occurring in the object position; conversely, each blank-node occurred on average 4.239 times in the subject position of a triple, with 69 thousand (0.04%) not occurring in the subject position.[5] Thus, we surmise that almost all blank-nodes appear in both the subject position and object position, but occur most prevalently in the former.

With respect to URIs, each had on average 9.41 data-level occurrences (1.8× the average for blank-nodes), with 4.399 average appearances in the subject position and 5.01 appearances in the object position—19.85 million (21.55%) did not appear in an object position, whilst 57.91 million (62.88%) did not appear in a subject position.

With respect to reuse across sources, each URI had a data-level occurrence in, on average, 4.7 documents, and 1.008 PLDs—56.2 million (61.02%) of URIs appeared in only one document, and 91.3 million (99.13%) only appeared in one PLD. Also, reuse of URIs across documents was heavily weighted in favour of use in the object position: URIs appeared in the subject position in, on average, 1.061 documents and 0.346 PLDs; for the object position of non-`rdf:type` triples, URIs occurred in, on average, 3.996 documents and 0.727 PLDs.

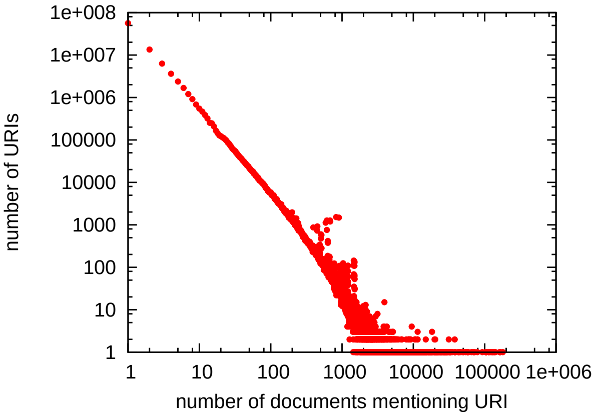
The URI with the most data-level occurrences (1.66 million) was `http://identi.ca/`, which refers to the homepage of an open-source micro-blogging platform, and which is commonly given as a value for `foaf:accountServiceHomepage`. The URI with the most reuse across documents (appearing in 179.3 thousand documents) was `http://creativecommons.org/licenses/by/3.0/`, which refers to the Creative Commons Attribution 3.0 licence, and which is commonly used (in various domains) as a value for various licensing properties, such as `dct:license`, `dc:rights`, `cc:license`, `mo:license`, `wrcc:licence`, etc. The URI with the most reuse across PLDs (appearing in 80 different domains) was `http://www.ldodds.com/foaf/foaf-a-matic`, which refers to an online application for generating FOAF profiles, and which featured most commonly as a value for `admin:generatorAgent` in such FOAF profiles. Although some URIs do enjoy widespread reuse across different documents and domains, in Figures 7.1 and 7.2 we give the distribution of reuse of URIs across documents and across PLDs, where a power-law relationship is roughly evident—again, the majority of URIs only appear in one document (61%) or in one PLD (99%).

From this analysis, we can conclude that with respect to data-level terms in our corpus:

- blank-nodes—which by their very nature cannot be reused across documents—are 1.8× more prevalent than URIs;
- despite a smaller number of unique URIs, each one is used in (probably coincidentally) on average 1.8× more triples than as for blank-nodes;
- unlike blank-nodes, URIs commonly only appear in either a subject position or an object position;
- each URI is reused in, on average, 4.7 documents, but usually only within the same domain—most external reuse is in the object position of a triple;

[4] It's worth noting that we may consolidate identifiers in terminological positions, such as the subject and/or object of `rdfs:subClassOf` relations. However, when needed, we *always* source terminological data from the *unconsolidated* corpus to ensure that it is unaffected by the consolidation process.

[5] We note that in RDF/XML syntax—essentially a tree-based syntax—unless `rdf:nodeID` is used, blank-nodes can only ever occur once in the object position of a triple, but can occur multiple times in the subject position.



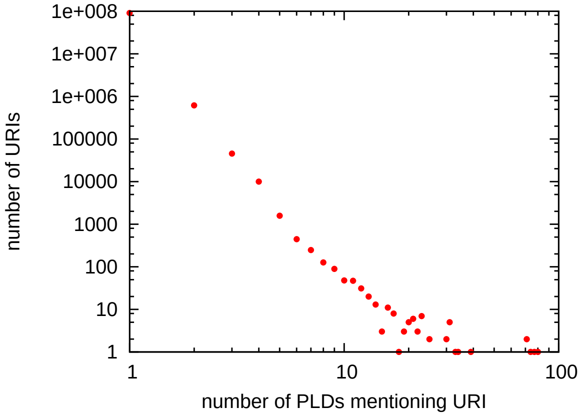


Figure 7.1: Distribution of URIs and the number of documents they appear in (in a data-position)

Figure 7.2: Distribution of URIs and the number of PLDs they appear in (in a data-position)

- 67.1% of URIs appear in only one document, and 99% of URIs appear in only one PLD.

We can conclude that within our Linked Data corpus, there is only sparse reuse of data-level terms across sources, and particularly across domains.

7.3 Base-line Consolidation

We now present the "base-line" algorithm for consolidation which leverages only those `owl:sameAs` relations which are explicitly asserted in the data.

7.3.1 High-level approach

The approach is straightforward:

1. scan the corpus and separate out all asserted `owl:sameAs` relations from the main body of the corpus;
2. load these relations into an in-memory index, which encodes the transitive and symmetric semantics of `owl:sameAs`;
3. for each equivalence class in the index, choose a canonical term;
4. scan the corpus again, canonicalising non-literal terms in the subject position or object position of a non-`rdf:type` triple.

Thus, we need only index a small subset of the corpus—11.93 million statements (1.1%) with the predicate `owl:sameAs`—and can apply consolidation by means of two scans. The non-trivial aspects of the algorithm are given by the in-memory equality index: we provide the details in Algorithm 7.1, where we use a map which stores a term (involved in a non-reflexive equality relation) as key, and stores a flat set of equivalent terms (of which the key is a member) as value—thus, we can perform a lookup of any term and retrieve the set of equivalent terms given by the `owl:sameAs` corpus.

With respect to choosing a canonical term, we prefer URIs over blank-nodes, thereafter choosing the term with the lowest lexical ordering:

Algorithm 7.1: Building equivalence map

```
Require: SameAs Data (On-Disk): SA
map := {}
for all $t \in$ SA do
  $Eq_s := \text{map.get}(t.s)$          /* $t.s$ denotes the subject of triple $t$ */
  if $Eq_s = \emptyset$ then
    $Eq_s := \{s\}$
  end if
  $Eq_o := \text{map.get}(t.o)$          /* $t.o$ denotes the object of triple $t$ */
  if $Eq_o = \emptyset$ then
    $Eq_o := \{o\}$
  end if
  if $Eq_s \neq Eq_o$ then
    $Eq_{s \cup o} := Eq_s \cup Eq_o$
    for $e \in Eq_{s \cup o}$ do
      $\text{map.put}(e, Eq_{s \cup o})$
    end for
  end if
end for
```

Definition 7.1 (Canonical Ordering) *Let $\leq_l$ denote the total lexical ordering defined (independently) over the set* U *and over the set* B*, and let $c_i, c_j \in \mathsf{U} \cup \mathsf{B}$. Now, we define the* canonical ordering *as a total order over the set* $\mathsf{U} \cup \mathsf{B}$*—denoted $\leq_c$—such that:*

$$c_i <_c c_j \Leftrightarrow (c_i \in \mathsf{U}, c_j \in \mathsf{B}) \vee \big((c_i, c_j \in \mathsf{U} \vee c_i, c_j \in \mathsf{B}) \wedge c_i <_l c_j\big)\,,$$
$$c_i =_c c_j \Leftrightarrow (c_i, c_j \in \mathsf{U} \vee c_i, c_j \in \mathsf{B}) \wedge c_i =_l c_j\,.$$

We then define the canonical function*—denoted by* can*—as:*

$$\mathsf{can} : 2^{\mathsf{U} \cup \mathsf{B}} \to \mathsf{U} \cup \mathsf{B}\,,$$
$$C \mapsto c \text{ s.t. } \forall c_i \in C : c_i \geq_c c\,,$$

which returns the lowest canonically-ordered element from a set of URIs and blank-nodes; we call the result the canonical identifier *of that set.*

We use this ordering to assign a canonical identifier to each equivalence class indexed by Algorithm 7.1. Once the equivalence index has been finalised, we rescan the corpus and canonicalise the data as per Algorithm 7.2. (Note that in practice, all Eq_x are pre-sorted according to $\leq_c$ such that to derive $\mathsf{can}(Eq_x)$, we need only poll the first element of the set.)

7.3.2 Distributed approach

Again, distribution of the approach is fairly straightforward, as follows:

1. **run**: scan the distributed corpus (split over the slave machines) in parallel to extract triples with `owl:sameAs` as predicate;
2. **gather**: gather all `owl:sameAs` relations onto the master machine, and build the in-memory equivalence map;
3. **flood/run**: send the equivalence map (in its entirety) to each slave machine, and apply the consolidation scan in parallel.

As we will see in the next section, the most expensive methods—involving the two scans of the main corpus—can be conducted in parallel.

Algorithm 7.2: Canonicalising input data

```
Require: Input Corpus (On-Disk): IN
Require: Output (On-Disk): OUT
Require: Equivalence Map: map                    /* from Algorithm 7.1 */
for all $t \in$ IN do
  $t' := t$
  $Eq_s :=$ map.get($t.s$)       /* $t.s, t.p, t.o$ denote subject, predicate & object of $t$ resp. */
  if $Eq_s \neq \emptyset$ then
    $t'.s :=$ can($Eq_s$)
  end if
  if $t.p \neq$ rdf:type $\wedge\ t.o \notin$ L then
    $Eq_o :=$ map.get($t.o$)
    if $Eq_o \neq \emptyset$ then
      $t'.o :=$ can($Eq_o$)
    end if
  end if
  output $t'$ to OUT
end for
```

7.3.3 Performance Evaluation

As per the introduction of this chapter, we apply consolidation over the raw, pre-distributed corpus (as directly produced by the crawler in Chapter 4). We again use eight slave machines and one master machine.

The entire consolidation process took 63.3 min, with the bulk of time taken as follows: the first scan extracting `owl:sameAs` statements took 12.5 min, with an average idle time for the servers of 11 s (1.4%)—i.e., on average, the slave machines spent 1.4% of the time idly waiting for peers to finish. Transferring, aggregating and loading the `owl:sameAs` statements on the master machine took 8.4 min. The second scan rewriting the data according to the canonical identifiers took in total 42.3 min, with an average idle time of 64.7 s (2.5%) for each machine at the end of the round. The slower time for the second round is attributable to the extra overhead of rewriting the data to disk, as opposed to just reading.

In Table 7.1, we give a breakdown of the timing for the tasks. Of course, please note that the percentages are a function of the number of machines where, e.g., a higher number of slave machines will correspond to a higher percentage of time on the master machine. However, independent of the number of slaves, we note that the master machine required 8.5 min for coordinating globally-required `owl:sameAs` knowledge, and that the rest of the task time is spent in embarrassingly parallel execution (amenable to reduction by increasing the number of machines). For our setup, the slave machines were kept busy for, on average, 84.6% of the total task time; of the idle time, 87% was spent waiting for the master to coordinate the `owl:sameAs` data, and 13% was spent waiting for peers to finish their task due to sub-optimal load balancing. The master machine spent 86.6% of the task idle waiting for the slaves to finish.

7.3.4 Results Evaluation

We extracted 11.93 million raw `owl:sameAs` statements, forming 2.16 million equivalence classes mentioning 5.75 million terms (6.24% of URIs)—an average of 2.65 elements per equivalence class. Of the 5.75 million terms, only 4,156 were blank-nodes. Figure 7.3 presents the distribution of sizes of the equivalence classes, where in particular we note that 1.6 million (74.1%) equivalence classes contain the minimum two equivalent identifiers.

Table 7.2 lists the canonical URIs for the largest 5 equivalence classes, where the largest class contained 8,481 equivalent terms; we also indicate whether or not the results were verified as correct/incorrect by

Category	**min**	**% Total**
Total execution time	63.3	100
Master (Local)		
Executing	8.5	13.4
Aggregating `owl:sameAs`	8.4	13.3
Miscellaneous	0.1	0.1
Idle (waiting for slaves)	54.8	86.6
Slave (Parallel)		
Avg. Executing (total)	53.5	84.6
Extract `owl:sameAs`	12.3	19.5
Consolidate	41.2	65.1
Avg. Idle	9.8	15.4
Waiting for peers	1.3	2
Waiting for master	8.5	13.4

Table 7.1: Breakdown of timing of distributed baseline consolidation

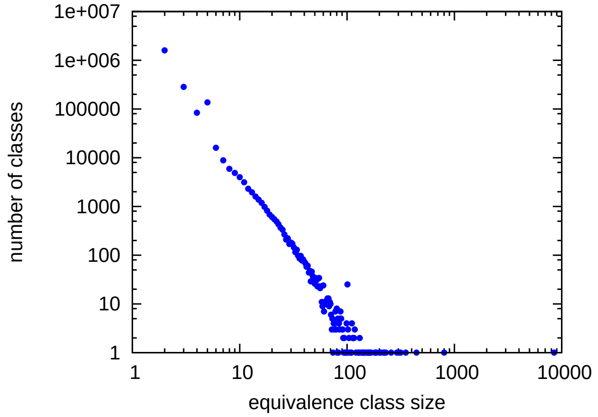


Figure 7.3: Distribution of sizes of equivalence classes on log/log scale

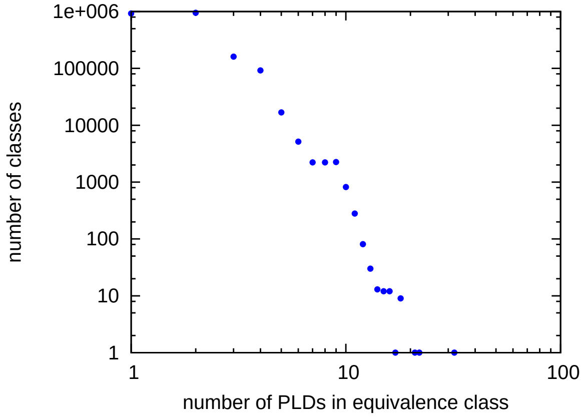


Figure 7.4: Distribution of the number of PLDs per equivalence class on log/log scale

manual inspection. Indeed, we deemed classes (1) and (2) to be incorrect, due to over-use of `owl:sameAs` for linking drug-related entities in the DailyMed and LinkedCT exporters. Results (3) and (5) were verified as correct consolidation of prominent Semantic Web related authors—respectively, `Dieter Fensel` and `Rudi Studer`—where authors are given many duplicate URIs by the RKBExplorer coreference index.[6] Result (4) contained URIs from various sites generally referring to the United States, mostly from DBPedia and LastFM. With respect to the DBPedia URIs, these (i) were equivalent but for capitilisation variations or stop-words, (ii) were variations of abbreviations or valid synonyms, (iii) were different language versions (e.g., `dbpedia:États_Unis`), (iv) were nicknames (e.g., `dbpedia:Yankee_land`), (v) were related but not equivalent (e.g., `dbpedia:American_Civilization`), (vi) were just noise (e.g., `dbpedia:LOL_Dean`).[7]

It is important to note that the largest equivalence classes are not a fair sample of the accuracy of

[6] For example, see the coreference results given by `http://www.rkbexplorer.com/sameAs/?uri=http://acm.rkbexplorer.com/id/person-53292-22877d02973d0d01e8f29c7113776e7e` (retr. 2010/09/14), which at the time of writing correspond to 436 out of the 443 equivalent URIs found for `Dieter Fensel`.

[7] Similar examples for problematic `owl:sameAs` relations from the DBPedia exporter are given by Halpin et al. [2010a].

#	Canonical Term (Lexically Lowest in Equivalence Class)	Size	OK?
1	`http://bio2rdf.org/dailymed_drugs:1000`	8,481	✗
2	`http://bio2rdf.org/dailymed_drugs:1042`	800	✗
3	`http://acm.rkbexplorer.com/id/person-53292-22877d02973d0d01e8f29c7113776e7e`	443	✓
4	`http://agame2teach.com/#ddb61cae0e083f705f65944cc3bb3968ce3f3ab59-ge_1`	353	✓/✗
5	`http://acm.rkbexplorer.com/id/person-236166-1b4ef5fdf4a5216256064c45a8923bc9`	316	✓

Table 7.2: Largest 5 equivalence classes

#	PLD	PLD	Relations
1	rdfize.com	uriburner.com	506,623
2	dbpedia.org	freebase.com	187,054
3	bio2rdf.org	purl.org	185,392
4	info:lc/authorities[8]	loc.gov	166,650
5	l3s.de	rkbexplorer.com	125,842
6	bibsonomy.org	l3s.de	125,832
7	bibsonomy.org	rkbexplorer.com	125,830
8	dbpedia.org	mpii.de	99,827
9	freebase.com	mpii.de	89,610
10	dbpedia.org	umbel.org	65,565

Table 7.3: Top 10 PLD pairs co-occurring in the equivalence classes, with number of equivalence classes they co-occur in

consolidation by explicit `owl:sameAs`: we conjecture that given some source of noise which incorrectly asserts coreference between individuals—for example, overly liberal use of `owl:sameAs`—the transitive and symmetric nature of equality will ensure that the problematic equivalence class "snowballs" and grows relatively large, drawing together all resources which are (possibly indirectly) connected by coreference due to such noise. Along these lines, we also randomly sampled 100 equivalent sets and manually checked for errors based on label (as an intuitive idea of what the identifier refers to) and type. We verified that all 100 were correct (or, more accurately, were not obviously incorrect).[9]

In Table 7.3, we give the most frequently co-occurring PLD-pairs in our equivalence classes, where datasets resident on these domains are "heavily" interlinked with `owl:sameAs` relations.

With respect to consolidation, identifiers in 78.6 million subject positions (7% of subject positions) and 23.2 million non-`rdf:type`-object positions (2.6%) were rewritten, giving a total of 101.9 million positions rewritten (5.1% of total rewritable positions). The average number of documents mentioning each URI rose slightly from 4.691 to 4.719 (a 0.6% increase) due to consolidation, and the average number of PLDs also rose slightly from 1.005 to 1.007 (a 0.2% increase).

7.4 Extended Reasoning Consolidation

We now look at extending the baseline, explicit `owl:sameAs` approach to include more expressive reasoning capabilities, using OWL 2 RL/RDF rules to infer novel `owl:sameAs` relations.

[9] Many were simple 'syntactic' equivalences from the `opiumfield.com` LastFM data exporter; for reference, we've published the 100 sets at `http://aidanhogan.com/swse/eqcs-sample-100.txt`.

Listing 7.3: Example of indirect inference of `owl:sameAs`

```
# From the FOAF Vocabulary:
 foaf:homepage rdfs:subPropertyOf foaf:isPrimaryTopicOf .
 foaf:isPrimaryTopicOf owl:inverseOf foaf:primaryTopic .
 foaf:isPrimaryTopicOf a owl:InverseFunctionalProperty .

# From Example Document A:
 exA:axel foaf:homepage <http://polleres.net/> .

# From Example Document B:
 <http://polleres.net/> foaf:primaryTopic exB:apolleres .

# Inferred through prp-spo1:
 exA:axel foaf:isPrimaryTopicOf <http://polleres.net/> .
# Inferred through prp-inv:
 exB:apolleres foaf:isPrimaryTopicOf <http://polleres.net/> .

# Subsequently, inferred through prp-ifp:
 exA:axel owl:sameAs exB:apolleres .
 exB:apolleres owl:sameAs exA:axel .
```

7.4.1 High-level approach

In Table B.8, we provide the pertinent rules for inferring new `owl:sameAs` relations from the data. However, after analysis of our corpus, we observed that no documents used the `owl:maxQualifiedCardinality` construct required for the cls-maxqc* rules, and that only one document defined one `owl:hasKey` axiom[10] involving properties with <5 occurrences as predicates in the data (cf. Table 5.2); given the rarity of these axioms, we leave implementation of these rules for future work and note that these new OWL 2 constructs have probably not yet had time to find proper traction on the Web. (Note also that the rules prp-key cls-maxqc3/cls-maxqc4 supporting these axioms do not have a single variable appearing in all A-atoms, and thus are not supportable through our current implementation which relies on merge-join operations.)

Thus, on top of inferencing involving explicit `owl:sameAs`, we also infer such relations through the following four rules, the set of which we denote as $\mathcal{O}2\mathcal{R}^{=}$:

1. prp-fp which supports the semantics of properties typed `owl:FunctionalProperty`;

2. prp-ifp which supports the semantics of properties typed `owl:InverseFunctionalProperty`;

3. cls-maxc2 which supports the semantics of classes with a specified cardinality of `1` for some defined property (a class restricted version of the functional-property inferencing); and

4. cls-exc2* which gives an exact cardinality version of cls-maxc2, but is not in OWL 2 RL/RDF.[11]

However, applying only these rules may lead to incomplete `owl:sameAs` inferences; for example, consider the data in Listing 7.3 where we we need OWL 2 RL/RDF rules prp-inv and prp-spo1—handling standard `owl:inverseOf` and `rdfs:subPropertyOf` inferencing respectively—to infer the `owl:sameAs` relation entailed by the data.

Thus, we also pre-apply more general OWL 2 RL/RDF reasoning over the corpus to derive more complete

[10] `http://huemer.lstadler.net/role/rh.rdf`; retr. 2010/11/27

[11] Exact cardinalities are disallowed in OWL 2 RL due to their effect on the formal proposition of completeness underlying the profile, but such considerations are moot in our scenario.

Algorithm 7.3: Extended consolidation approach: overview

```
Require: INPUT CORPUS: IN                                          /* on-disk input */
Require: OUTPUT: OUT                        /* output with canonicalised subjs. and objs. */
$TBox := \mathsf{TBox}(\mathcal{O}2\mathcal{R}^- \cup \mathcal{O}2\mathcal{R}^=, \mathtt{IN})$     /* extract T-Box (including sources): see § 5.4 */
$P^{A+} := \widehat{\mathsf{Ground}}^T(\mathcal{O}2\mathcal{R}^-, TBox)$                /* auth. T-ground rules: see § 5.4.2 */
$P^{FP} := \widehat{\mathsf{Ground}}^T(\{\mathsf{prp\text{-}fp}\}, TBox)$                /* auth. T-ground rules: see § 5.4.2 */
$P^{IFP} := \widehat{\mathsf{Ground}}^T(\{\mathsf{prp\text{-}ifp}\}, TBox)$              /* auth. T-ground rules: see § 5.4.2 */
$P^{card} := \widehat{\mathsf{Ground}}^T(\{\mathsf{cax\text{-}maxc2}, \mathsf{cax\text{-}exc2*}\}, TBox)$   /* auth. T-ground rules: see § 5.4.2 */
$\mathtt{TMP}_0^{FP}, \mathtt{TMP}_0^{IFP}, \mathtt{TMP}_0^{card}, \mathtt{TMP}_0^{sA} := \{\}$
for all $t \in \mathtt{IN}$ do
  $I := \mathsf{lm}(P^{A+} \cup \{t\})$        /* get inferences for triple wrt. $P^{TA}$: see § 5.2 (note $t \in I$) */
  for all $t' \in I$ do
    if $t'.p =$owl:sameAs then                             /* if predicate is owl:sameAs */
      write $t'$ to $\mathtt{TMP}_0^{sA}$
    end if
    if $\exists R \in P^{FP}, \exists A \in \mathsf{ABody}(R)$ s.t. $A \triangleright t'$ then
      write $t'$ to $\mathtt{TMP}_0^{FP}$
    end if
    if $\exists R \in P^{IFP}, \exists A \in \mathsf{ABody}(R)$ s.t. $A \triangleright t'$ then
      write $t'$ to $\mathtt{TMP}_0^{IFP}$
    end if
    if $\exists R \in P^{card}, \exists A \in \mathsf{ABody}(R)$ s.t. $A \triangleright t'$ then
      write $t'$ to $\mathtt{TMP}_0^{card}$
    end if
  end for
end for
$novel := \mathtt{TMP}_0^{sA} \neq \{\} \vee \mathtt{TMP}_0^{FP} \neq \{\} \vee \mathtt{TMP}_0^{IFP} \neq \{\} \vee \mathtt{TMP}_0^{card} \neq \{\}$
while $novel$ do
  compute owl:sameAs from $\mathtt{TMP}_i^{FP}$, write to $\mathtt{TMP}_i^{sA}$                /* see Alg. 7.5 */
  compute owl:sameAs from $\mathtt{TMP}_i^{IFP}$, write to $\mathtt{TMP}_i^{sA}$               /* see Alg. 7.6 */
  compute owl:sameAs from $\mathtt{TMP}_i^{card}$, write to $\mathtt{TMP}_i^{sA}$              /* see Alg. 7.7 */
  compute owl:sameAs closure $\mathtt{TMP}_i^{sA}$, write to $\mathtt{TMP}_{i+1}^{sA}$            /* see Alg. 7.8 */
  $novel := \mathtt{TMP}_i^{sA} \neq \mathtt{TMP}_{i+1}^{sA}$
  $i{+}{+}$
  if $novel$ then
    rewrite subjs. of $\mathtt{TMP}_i^{FP}, \mathtt{TMP}_i^{card}$, objs. of $\mathtt{TMP}_i^{IFP}$, by $\mathtt{TMP}_i^{sA}$     /* see Alg. 7.9 */
  end if
end while
rewrite subjs., objs. of IN by $\mathtt{TMP}_i^{sA}$                                /* see Alg. 7.9 */
```

owl:sameAs results; in particular, we additionally apply the subset of inference rules from the A-linear $\mathcal{O}2\mathcal{R}^{\propto\mathsf{A}}$ profile (§ 5.4.1) which deals with assertional reasoning, and which is listed in Table B.4—we denote this subset of $\mathcal{O}2\mathcal{R}^{\propto\mathsf{A}}$ as $\mathcal{O}2\mathcal{R}^-$:

$$\mathcal{O}2\mathcal{R}^- := \{R \in \mathcal{O}2\mathcal{R}^{\propto\mathsf{A}} : |\mathsf{ABody}(R)| = 1\}$$

Note that we also exclude rule eq-sym from $\mathcal{O}2\mathcal{R}^-$, where the semantics of equality (including symmetry and transitivity) are supported in a bespoke, optimised manner later in this section. Finally, note that we again apply authoritative reasoning (§ 5.4.2).

Continuing, Algorithm 7.3 outlines our extended consolidation process, where the high-level approach is as follows:

1. extract all terminological triples from the corpus which are an instance of a T-atom from the body of

a rule in $\mathcal{O}2\mathcal{R}^{=} \cup \mathcal{O}2\mathcal{R}^{-}$ (Line 1, Algorithm 7.3);

2. use these data to ground the terminological atoms in the $\mathcal{O}2\mathcal{R}^{=} \cup \mathcal{O}2\mathcal{R}^{-}$ rules, creating a larger set of partially evaluated, assertional rules (Lines 1–5, Algorithm 7.3);

3. apply reasoning using the partially evaluated $\mathcal{O}2\mathcal{R}^{-}$ rules over the corpus (as per § 5.2), and output any input or inferred triples which have either (i) `owl:sameAs` as predicate; or (ii) are an instance of an atom in the body of a partially evaluated rule from $\mathcal{O}2\mathcal{R}^{=}$ (Lines 7–23, Algorithm 7.3);

4. compute the least fixpoint of `owl:sameAs` statements from the *consolidation-relevant data* extract in the previous step (Lines 25–35, Algorithm 7.3);

5. apply consolidation over the main corpus, canonicalising identifiers with respect to the closed `owl:sameAs` data (Line 36).

In Step 1, we extract terminological data required for application of both the general and consolidation rules (in the lingo of § 5.2, this is the T-Box of $\mathcal{O}2\mathcal{R}^{=} \cup \mathcal{O}2\mathcal{R}^{-}$ extracted from the corpus). Subsequently, in Step 2, we produce authoritative T-ground rules by partially evaluating rules in $\mathcal{O}2\mathcal{R}^{=} \cup \mathcal{O}2\mathcal{R}^{-}$ with respect to the extracted terminological knowledge (§ 5.2, § 6.2.2). We now briefly illustrate these two steps for two rules in $\mathcal{O}2\mathcal{R}^{=}$ by way of example:

Example 7.1 *Take the rule* prp-ifp*:*

$$(\texttt{?x}_1,\ \texttt{owl:sameAs},\ \texttt{?x}_2) \leftarrow \underline{(\texttt{?p},\ \texttt{a},\ \texttt{owl:InverseFunctionalProperty})},\ (\texttt{?x}_1,\ \texttt{?p},\ \texttt{?y}),(\texttt{?x}_2,\ \texttt{?p},\ \texttt{?y})$$

where the T-atom in the body is underlined. Now take the terminological axiom (from the `foaf:` *vocabulary):*

$$(\texttt{foaf:mbox},\ \texttt{a},\ \texttt{owl:InverseFunctionalProperty})$$

This axiom is extracted during Step 1 above since it is an instance of the T-atom of prp-ifp $\in \mathcal{O}2\mathcal{R}^{=}$*. Now, the above axiom must be served authoritatively for* `foaf:mbox`*—i.e., must be served in the document given by* redirs(`foaf:mbox`)*—to be considered for consolidation; this is because in the T-atom of the above rule, the only variable which also appears in an A-atom is* `?p`*, and* `foaf:mbox` *is substituted for this variable (§ 6.2.2). Since the axiom is served by the FOAF vocabulary—which corresponds to* redirs(`foaf:mbox`)*—Step 2 will generate the following authoritative T-ground rule:*

$$(\texttt{?x}_1,\ \texttt{owl:sameAs},\ \texttt{?x}_2) \leftarrow (\texttt{?x}_1,\ \texttt{foaf:mbox},\ \texttt{?y}),(\texttt{?x}_2,\ \texttt{foaf:mbox},\ \texttt{?y})\ .$$

Taking a different example, consider rule cls-maxc2*:*

$$(\texttt{?y}_1,\ \texttt{owl:sameAs},\ \texttt{?y}_2) \leftarrow \underline{(\texttt{?x},\ \texttt{owl:maxCardinality},\ \texttt{1}),(\texttt{?x},\ \texttt{owl:onProperty},\ \texttt{?p})},\ (\texttt{?u},\ \texttt{?p},\ \texttt{?y}_1),(\texttt{?u},\ \texttt{?p},\ \texttt{?y}_2),(\texttt{?u},\ \texttt{a},\ \texttt{?x})$$

and the axiom (from the `like:` *vocabulary):*

$$(\texttt{\_:x},\ \texttt{owl:maxCardinality},\ \texttt{1}),(\texttt{\_:x},\ \texttt{owl:onProperty},\ \texttt{rev:rating})$$

where both triples will be extracted during Step 1. Here, the above axiom must be served authoritatively for `_:x` *or* `rev:rating`*; since every document is authoritative for its blank nodes, this axiom must necessarily be authoritative.*[12] *Thus, the authoritative T-ground rule produced in Step 2 will be:*

$$(\texttt{?y}_1,\ \texttt{owl:sameAs},\ \texttt{?y}_2) \leftarrow (\texttt{?u},\ \texttt{rev:rating},\ \texttt{?y}_1),(\texttt{?u},\ \texttt{rev:rating},\ \texttt{?y}_2),(\texttt{?u},\ \texttt{a},\ \texttt{\_:x})\ .$$

[12] If `_:x` were replaced with a URI `ex:x`, the axiom would have to be defined in the document given by either redirs(`ex:x`) *or* redirs(`rev:rating`) to be authoritative.

Algorithm 7.4: Write equivalence-class to output

```
Require: EQUIVALENCE CLASS: $Eq$                          /* in-memory */
Require: SAMEAS OUTPUT: SA_OUT                            /* on-disk */
$can := \mathsf{can}(Eq)$
for all $c \in Eq, c \neq can$ do
    write $(can, \texttt{owl:sameAs}, c)$ to SA_OUT
end for
```

These T-ground rules are then used to find consolidation-relevant data: triples which can serve as instances of the atoms in the respective heads (note again that `_:x` *is a skolem constant [§ 3.1]).* ◊

In Step 3, we apply the T-ground $\mathcal{O}2\mathcal{R}^-$ rules over the corpus (as per Algorithm 5.1), where any input or inferred statements that are an instance of a body atom from a T-ground $\mathcal{O}2\mathcal{R}^=$ rule—e.g., triples which match (`?x`, `foaf:mbox`, `?y`), (`?u`, `rev:rating`, `?y`), or (`?u`, `a`, `_:x`) from Example 7.1—are buffered to a separate file, including any `owl:sameAs` statements found. (Note that during $\mathcal{O}2\mathcal{R}^-$ inferencing, we use the rule-indexing and merging optimisations as described in § 5.3, and we discard inferred data which is not relevant for consolidation.) We thus extract a focussed sub-corpus from which `owl:sameAs` relations can be directly derived using the $\mathcal{O}2\mathcal{R}^=$ inference rules.

Subsequently, in Step 4, we must now compute the *canonicalised closure* of the `owl:sameAs` statements. For the baseline consolidation approach presented in § 7.3 we used an in-memory equality index to support the semantics of `owl:sameAs`, to represent the equivalence classes and chosen canonical terms, and to provide the lookups required during canonicalisation of the corpus. However, by using such an approach, the scalability of the system is bound by the memory resources of the hardware (which itself cannot be solved by distribution since—in our approach—all machines require knowledge about all same-as statements). In particular, the extended reasoning approach will produce a large set of such statements which will require a prohibitive amount of memory to store.[13] Thus, we turn to on-disk methods to handle the transitive and symmetric closure of the `owl:sameAs` corpus and to perform the subsequent consolidation of the corpus in Step 5.

In particular, following the same rationale as the previous implementations detailed in this thesis, we mainly rely upon the following three on-disk primitives: (i) *sequential scans* of flat files containing line-delimited tuples;[14] (ii) *external-sorts* where batches of statements are sorted in memory, the sorted batches written to disk, and the sorted batches merged to the final output; and (iii) *merge-joins* where multiple sets of data are sorted according to their required join position, and subsequently scanned in an interleaving manner which aligns on the join position and where an in-memory join is applied for each individual join element. Using these primitives to perform the `owl:sameAs` computation minimises the amount of main memory required [Hogan et al., 2007a, 2009b].

First, the assertional data specifically relevant for prp-fp (functional properties), prp-ifp inverse-functional properties and cax-maxc2/cax-exc2* (cardinality-of-one restrictions) are written to three separate on-disk files. Any `owl:sameAs` data produced directly by Step 3 are written to a fourth file. We then apply inferencing over the first three files.

For functional-property and cardinality reasoning, a consistent join variable for the assertional body atoms is given by the subject position; for inverse-functional-property reasoning, a join variable is given

[13] Currently, we store entire (uncompressed) strings in memory, using a flyweight pattern (interning) which guarantees unique references. In future, we may consider lossless string compression techniques over the repetitive URI strings (e.g., see [Michel et al., 2000; Fernández et al., 2010]) to increase the in-memory capacity.

[14] These files are G-Zip compressed flat files of N-Triple–like syntax encoding arbitrary length tuples of RDF constants.

Algorithm 7.5: Computing prp-fp inferences

```
Require: prp-fp-INPUT: FP_IN              /* on-disk input triples sorted lexicographically */
Require: SAMEAS OUTPUT: SA_OUT                                          /* on-disk output */
 1: sort FP_IN by lexicographical ($s$–$p$–$o$) order
 2: $Eq_{FP} := \{\}$; $i := 0$
 3: for all $t_i \in$ FP_IN do
 4:   if $i \neq 0 \wedge (t_i.s \neq t_{i-1}.s \vee t_i.p \neq t_{i-1}.p)$ then
 5:     if $|Eq_{FP}| \geq 2$ then
 6:       write $Eq_{FP}$ to SA_OUT                            /* as per Algorithm 7.4 */
 7:     end if
 8:     $Eq_{FP} := \{\}$
 9:   end if
10:   if $t_i.o \notin \mathsf{L}$ then
11:     $Eq_{FP} := Eq_{FP} \cup \{t_i.o\}$
12:   end if
13:   $i$++
14: end for
15: repeat Lines 5–7 for final $Eq_{FP}$
```

Algorithm 7.6: Computing prp-ifp inferences

```
Require: prp-ifp-INPUT: IFP_IN                                    /* on-disk input triples */
Require: SAMEAS OUTPUT: SA_OUT                                          /* on-disk output */
 1: sort IFP_IN by inverse ($o$–$p$–$s$) order
 2: $Eq_{IFP} := \{\}$; $i := 0$
 3: for all $t_i \in$ IFP_IN do
 4:   if $i \neq 0 \wedge (t_i.o \neq t_{i-1}.o \vee t_i.p \neq t_{i-1}.p)$ then
 5:     if $|Eq_{IFP}| \geq 2$ then
 6:       write $Eq_{IFP}$ to SA_OUT                           /* as per Algorithm 7.4 */
 7:     end if
 8:     $Eq_{IFP} := \{\}$
 9:   end if
10:   $Eq_{IFP} := Eq_{IFP} \cup \{t_i.s\}$
11:   $i$++
12: end for
13: repeat Lines 5–7 for final $Eq_{IFP}$
```

by the object position.[15] Thus, we can sort the former sets of data according to subject and perform a merge-join by means of a linear scan thereafter; the same procedure applies to the latter file, sorting and merge-joining on the object position. Applying merge-join scans, we produce new `owl:sameAs` statements. These techniques are detailed in Algorithm 7.5 for functional-property inferences, Algorithm 7.6 for inverse-functional property inferences, and Algorithm 7.7 for cardinality-of-one inferences; note that when writing `owl:sameAs` inferences to disk, we write results in a canonical form, briefly outlined in Algorithm 7.4. Also note that Algorithm 7.7 requires an in-memory map (denoted $\overset{p\mapsto C}{\text{map}}$) which maps from properties to the set of cardinality-of-one restrictions it is associated with; i.e.:

$$\overset{p\mapsto C}{\text{map}} : \mathsf{U} \to 2^{\mathsf{U}\cup\mathsf{B}} ,$$
$$\mathtt{p} \mapsto \{\mathtt{c}|(\mathtt{c},\ \mathtt{owl{:}onProperty},\ \mathtt{p}),\ (\mathtt{c},\ \mathtt{owl{:}[maxC/c]ardinality},\ 1) \in TBox\} .$$

[15] Although a predicate-position join is also available, we prefer data-position joins which provide smaller batches of data for the in-memory join.

Algorithm 7.7: Computing cax-maxc2/cax-exc2* inferences

```
Require: cax-maxc2/cax-exc2*-INPUT: CARD_IN                      /* on-disk input triples */
Require: SAMEAS OUTPUT: SA_OUT                                    /* on-disk output */
Require: $\overset{p\mapsto C}{\text{map}}$      /* maps a property to its resp. set of (max/exact) cardinality-of-one classes */
 1: sort CARD_IN by lexicographical ($s$–$p$–$o$) order
 2: $Eq_{Card} := \{\}$; $i := 0$
 3: $\overset{p\mapsto O}{\text{map}} := \{\}$      /* maps a property to its set of objects for the resp. subject */
 4: $O := \{\}$                                                   /* current objects */
 5: for all $t_i \in$ CARD_IN do
 6:   if $i \neq 0 \wedge (t_i.s \neq t_{i-1}.s \vee t_i.p \neq t_{i-1}.p)$ then
 7:     $\overset{p\mapsto O}{\text{map}}$.put$(p, O)$
 8:     if $t_i.s \neq t_{i-1}.s$ then
 9:       for all $(p, O) \in \overset{p\mapsto O}{\text{map}}$ do
10:         if $\overset{p\mapsto C}{\text{map}}$.get$(p) \cap \overset{p\mapsto O}{\text{map}}.get($rdf:type$) \neq \{\}$ then
11:           write $O$ to SA_OUT                                 /* as per Algorithm 7.4 */
12:         end if
13:       end for
14:       $\overset{p\mapsto O}{\text{map}} := \{\}$
15:     end if
16:     $O := \{\}$
17:   end if
18:   if $t_i.o \notin$ L then
19:     $O := O \cup \{t_i.o\}$
20:   end if
21:   $i$++
22: end for
23: repeat Lines 9–13 for final $\overset{p\mapsto O}{\text{map}}$
```

Both the originally asserted and newly inferred `owl:sameAs` relations are similarly written to an on-disk file, over which we now wish to perform the canonicalised symmetric/transitive closure. We apply a similar method again—using sorts, merge-joins and scans—where the technique is detailed in Algorithm 7.8. Here, we briefly sketch the procedure, where we use the canonical ordering $\leq_c$ from Definition 7.1, we use `sA` as a shortcut for `owl:sameAs`, and use e_x to denote members of $\mathsf{U} \cup \mathsf{B}$ such that $e_i <_c e_j \Leftrightarrow i < j$. The process is as follows:

1. we only materialise symmetric equality relations which involve a (possibly intermediary) canonical term chosen by a lexical ordering: given e_2 `sA` e_1, we materialise e_1 `sA` e_2; given e_1 `sA` e_2 `sA` e_3, we materialise the relations e_1 `sA` e_2, e_1 `sA` e_3, and their inverses, but do not materialise e_2 `sA` e_3 or its inverse;

2. a compressed (canonical) form of the transitive/symmetric closure is supported by iterative merge-join scans:

   - in each scan, we load all `sA` information for each group of triples with the same subject (grouped by the sort):
     - if we find, e.g., e_1 `sA` e_2, e_1 `sA` e_3, ...—i.e., the subject is lower than all attached identifiers (the subject is the canonical identifier)—we infer e_2 `sA` e_1, e_3 `sA` e_1, ...; these inferences are considered *repeated*;
     - if we find, e.g., e_3 `sA` e_2, e_3 `sA` e_1, ...—i.e., the subject is higher than all attached identifiers—we do nothing;

Algorithm 7.8: Computing compressed `owl:sameAs` closure

```
Require: owl:sameAs-INPUT: SA_IN        /* on-disk input triples with the predicate owl:sameAs */
Require: SAMEAS OUTPUT: SA_OUT                                          /* on-disk output */
 1: $r := 0$; $\texttt{SA\_TMP}_r := \{\}$
 2: if $\texttt{SA\_IN} \neq \{\}$ then
 3:    for all $t \in \texttt{SA\_IN}$ do
 4:       write $t$ and $(t.o, \texttt{owl:sameAs}, t.s)$ to $\texttt{SA\_TMP}_r$      /* write triples and their inverses */
 5:    end for
 6:    $end :=$ false;
 7: end if
 8: while $!end$ do
 9:    sort $\texttt{SA\_TMP}_r$ by lexicographical $(s{-}p{-}o)$ order
10:    $\texttt{SA\_TMP}_{r+1} := \{\}$; $Eq_{sA} := \{\}$; $i := 0$;
11:    $end :=$ true
12:    for all $t_i \in \texttt{SA\_TMP}_r$ do
13:       if $i \neq 0 \wedge t_i.s \neq t_{i-1}.s$ then
14:          if $\exists e \in Eq_{sA}$ s.t. $e >_c t_{i-1}.s$ then
15:             $can := \mathsf{can}(Eq_{sA} \cup \{t_{i-1}.s\})$
16:             if $can \neq t_{i-1}.s$ then
17:                $end :=$ false
18:             end if
19:             for all $c \in Eq_{sA} \cup \{t_{i-1}.s\}$ s.t. $c \neq can$ do
20:                write $(can, \texttt{owl:sameAs}, c)$, $(c, \texttt{owl:sameAs}, can)$ to $\texttt{SA\_TMP}_{r+1}$
21:             end for
22:          end if
23:          $Eq_{sA} := \{\}$;
24:       end if
25:       $Eq_{sA} := Eq_{sA} \cup \{t_i.o\}$
26:       $i{+}{+}$
27:    end for
28:    repeat Lines 14–22 for final $Eq_{sA}$ & $t_{i-1}.s$
29:    $r{+}{+}$
30: end while
31: $\texttt{SA\_OUT} := \texttt{SA\_TMP}_r$
```

  - if we find, e.g., e_3 `sA` e_1, e_3 `sA` e_2, e_3 `sA` e_4—i.e., the subject is not the canonical identifier, but *is* lower than some attached identifiers—we infer e_1 `sA` e_2, e_1 `sA` e_3, e_1 `sA` e_4, and the inverses e_2 `sA` e_1, e_3 `sA` e_1, e_4 `sA` e_1; these inferences are considered *novel*;

- at the end of the scan, the data output by the previous scan are sorted;
- the process is then iterative: in the next scan, if we find e_4 `sA` e_1 and e_4 `sA` e_5, we infer e_1 `sA` e_5 and e_5 `sA` e_1;, etc.;

3. the above iterations stop when a fixpoint is reached and no *novel* inferences are given.

Intuitively, each iteration translates two-hop ($a \overset{\texttt{sA}}{\leftrightarrow} b \overset{\texttt{sA}}{\leftrightarrow} c$) equivalences into one-hop canonical equivalences ($a \overset{\texttt{sA}}{\leftrightarrow} b, a \overset{\texttt{sA}}{\leftrightarrow}, c$). The eventual result of this process is a set of canonicalised equality relations representing the symmetric/transitive closure of `owl:sameAs` relations. Note that we implement some optimisations on top of this process (for clarity, these are omitted from Algorithm 7.8):

- we leave the predicate `owl:sameAs` implicit, and only handle pairs of identifiers;
- we write *repeated inferences* (but not the inverses) to a separate file: only novel data (and inverses of repeated inferences) need to be sorted at the end of each scan, where these data can be subsequently

Algorithm 7.9: Canonicalising data using on-disk `owl:sameAs` closure

```
Require: owl:sameAs-INPUT: SA_IN      /* canonicalised owl:sameAs closure (as given by Alg. 7.8) */
Require: DATA INPUT: DATA_IN                        /* on-disk data to be canonicalised */
Require: CAN. DATA OUTPUT: CDATA_OUT             /* on-disk output for canonicalised data */
Require: POSITIONS: $Pos$      /* positions to canonicalise (e.g. $\{0,2\}$ for RDF sub. & obj.) */
if $\texttt{SA\_IN} \neq \{\} \wedge Pos \neq \{\} \wedge \texttt{DATA\_IN} \neq \{\}$ then
  $\texttt{SA\_IN}^{-} := \{\}$; $i := 0$
  for all $t \in \texttt{SA\_IN}$, $t.s >_c t.o$ do
    write $t$ to $\texttt{SA\_IN}^{-}$                /* only write tuples with non-canonical subject */
  end for
  $\texttt{CDATA\_OUT}_i := \texttt{DATA\_IN}$
  for all $pos_i \in Pos$ do
    sort $\texttt{CDATA\_OUT}_i$ to $\texttt{CDATA\_OUT}_i^s$ by $(pos_i, \ldots)$ order
    for all $t \in Pos$ do
      $t' := t$
      if $\exists (t.pos_i, \texttt{owl:sameAs}, can) \in \texttt{SA\_IN}^{-}$ then   /* by external merge-join with $\texttt{SA\_IN}^{-}$ */
        $t'.pos_i := can$
      end if
      write $t'$ to $\texttt{CDATA\_OUT}_{i+1}$
    end for
    $i$++
  end for
  $\texttt{CDATA\_OUT} := \texttt{CDATA\_OUT}_{i+1}$
else
  $\texttt{CDATA\_OUT} := \texttt{DATA\_IN}$
end if
```

merge-sorted with the separate repeated-inferences file (which are inherently sorted);

- we use a fixed size, in-memory equivalence map—as described in Algorithm 7.1—as a cache, to store partial equivalence "chains" and thus accelerate the fixpoint.

With respect to the last item, for each scan, we fill a fresh, in-memory equivalence map until the main-memory capacity is reached: on the first scan, we attempt to load all data, whereas on subsequent scans, we only attempt to load novel inferences found. When the capacity of the map is reached, we output the in-memory equivalences in canonical form (including inverses) and finish the scan using the standard on-disk merge-join operation, but where we also consult the map at each stage to see if a better (i.e., lower) canonical identifier is available therein. Note that if all data fit in the map on the first scan, then we need not apply the iterative process. Otherwise, the in-memory map accelerates the fixpoint, in particular by computing the small number of long equality chains which would otherwise require sorts and merge-joins over all of the canonical `owl:sameAs` data currently derived, and where the number of iterations would otherwise be $\log(n)$ where n is the length of the longest chain.

Now, we briefly describe the process of canonicalising data with respect to this on-disk equality corpus, where we again use sorts and merge-joins: the procedure is detailed in Algorithm 7.9. First, we prune the `owl:sameAs` index to only maintain a (lexicographically) sorted batch of relations s_2 `SA` s_1 such that $s_2 >_c s_1$—thus, given s_2 `SA` s_1, we know that s_1 is the canonical identifier, and s_2 is to be rewritten. We then sort the data according to the position which we wish to rewrite, and perform a merge-join over both the sorted data and the `owl:sameAs` file—this allows us to find canonical identifiers for the terms in the join position of the input, and to canonicalise these terms, buffering the (possibly rewritten) triple to an output file. If we want to rewrite multiple positions of a file of tuples (e.g., subject and object), we must rewrite one position, sort the intermediary results by the second position, and subsequently rewrite the second position.[16]

Note that in the derivation of owl:sameAs from the consolidation rules $\mathcal{O}2\mathcal{R}^{=}$, the overall process may be iterative. For instance, consider the data in Listing 7.4 from which the conclusion that exA:Axel is the same as exB:apolleres holds, but requires recursive application of rules: we see that new owl:sameAs relations (either asserted or derived from the consolidation rules) may in turn "align" terms in the join position of the consolidation rules, leading to new equivalences.

Listing 7.4: Example requiring recursive equality reasoning

```
exA:Axel foaf:isPrimaryTopicOf <http://polleres.net/> .
exB:apolleres foaf:isPrimaryTopicOf <http://axel.deri.ie/> .
<http://polleres.net/> owl:sameAs <http://axel.deri.ie/> .
```

Thus, for deriving the final owl:sameAs, we require a higher-level iterative process as follows (also given by Lines 25–35, Algorithm 7.3):

1. initially apply the consolidation rules, and append the results to a file alongside the owl:sameAs statements found in the input and from application of $\mathcal{O}2\mathcal{R}^{-}$ rules;
2. apply the initial closure of the aggregated owl:sameAs data collected thus far;
3. then, iteratively until no new owl:sameAs inferences are found:
   - canonicalise the identifiers in the join positions of the on-disk files containing the data for each consolidation rule according to the current owl:sameAs data;
   - derive new owl:sameAs inferences possible through the previous rewriting for each consolidation rule;
   - re-derive the closure of the owl:sameAs data including the new inferences.

Note that in the above iterative process, at each step we mark the delta given by newly rewritten or inferred statements, and only consider those inference steps which involve some part of the delta as novel: for brevity, we leave this implicit in the various algorithms.

The final closed file of owl:sameAs data can then be reused to rewrite the main corpus in two sorts and merge-join scans over subject and object, following the procedure outlined in Algorithm 7.9—again, note that we do not rewrite literals, predicates, or values of rdf:type (see § 7.1). Also, we maintain the original identifiers appearing in the corpus, outputting sextuples of the form:

$$(s, p, o, c, s', o')$$

where (s, p, o, c) are the quadruples containing possibly canonicalised s and o, and where s' and o' are the original identifiers found in the raw corpus.[17] This is not only useful for the repair step sketched in § 7.6, but

[16] One could consider instead building an on-disk map for equivalence classes and canonical identifiers and follow a consolidation procedure similar to the previous section over the unordered corpus: however, we would expect that such an on-disk index would have a low cache hit-rate given the nature of the data, which would lead to a high number of disk seek operations. An alternative approach might be to split and hash the corpus according to subject/object and split the equality data into relevant segments loadable in-memory on each machine: however, this would again require a non-trivial minimum amount of memory to be available over the given cluster.

[17] We use syntactic shortcuts in our file to denote when $s = s'$ and/or $o = o'$. Maintaining the additional rewrite information during the consolidation process is trivial, where the output of consolidating subjects gives quintuples (s, p, o, c, s'), which are then sorted and consolidated by o to produce the given sextuples.

also potentially useful for consumers of the consolidated data, who may, for example, wish to revert certain subsets of coreference flagged by users as incorrect.

Finally, we make some remarks with respect to incompleteness, where herein we are interested in deriving complete owl:sameAs results not involving literals or blacklisted data. Given that we are deliberately incomplete—e.g., that we do not materialise inferences which do not affect consolidation, and that we do not support rule eq-rep-p—we are more so interested in how we are *in*deliberately incomplete with respect to the derivation of owl:sameAs relations. In particular, we note that recursively applying the entire process again (as per Algorithm 7.3) over the output may lead to the derivation of new equivalences: i.e., we have not reached a fixpoint, and we may find new equivalences in subsequent applications of the consolidation process.

First, following from the discussion of § 5.2, equivalences which affect the Herbrand universe of the terminology of the data—i.e., the set of RDF constants appearing in some terminological triples—may cause incompleteness in our T-split approach. Aside from the cases of incompleteness provided in Example 5.4—where we are now also *deliberately* incomplete with respect to the case involving triples (1_a–9_a) since we do not support eq-rep-p—we demonstrate a novel example of how incompleteness can occur.

Example 7.2 *Take the following terminological axioms:*

1. (_:woman, owl:hasValue, ex:female)
2. (_:woman, owl:onProperty, ex:gender)
3. (_:woman, rdfs:subClassOf, _:person)
4. (_:person, owl:maxCardinality, 1)
5. (_:person, owl:onProperty, ex:gender)

along with the following assertional data:

6. (ex:female, owl:sameAs, ex:baineann)
7. (ex:Marie, ex:gender, ex:baineann)
8. (ex:Marie, ex:gender, ex:femme)

where, by eq-rep-o/eq-sym *we can infer:*

9. (ex:Marie, ex:gender, ex:female)

Now, by rules cls-hv2 *and* cax-sco *respectively, we should infer:*

10. (ex:Marie, a, _:woman)
11. (ex:Marie, a, _:person)

but we miss these inferences since cls-hv2 *is not applied over the consolidated data, and in any case, our canonicalisation would select* ex:baineann *over* ex:female, *where only the latter constant would allow for unification with the terminological axiom. Further, note that from triples* (7), (8) *&* (11) *and rule* cax-maxc2, *we should also infer:*

12. (ex:baineann, owl:sameAs, ex:femme)
13. (ex:femme, owl:sameAs, ex:baineann)

where we miss these owl:sameAs *relations by missing triples* (10) *&* (11). *Note (i) that if we were to rerun the consolidation process, we would find these latter equivalences in the second pass, and (ii) the equivalences in this example only involve* assertional identifiers *and not any members of a meta-class—more specifically, we have an equivalence involving an assertional identifier which appears in the terminology as a value for the* owl:hasValue *meta-property.* ◊

Aside from equivalences involving identifiers in the terminology of the corpus, we may also experience incompleteness if a given variable appears twice in the A-atom(s) of a rule in $\mathcal{O}2\mathcal{R}^-$. We now present such an example of incompleteness for a rule with a single assertional atom—one could imagine similar examples for rules with multiple assertional atoms.

Example 7.3 *Let* prp-hs2 *denote a rule which partially axiomatises the semantics of* `owl:hasSelf` *as follows:*

$$(\texttt{?u}, \texttt{a}, \texttt{?x}) \leftarrow \underline{(\texttt{x}, \texttt{owl:hasSelf}, \texttt{true}),(\texttt{x}, \texttt{owl:onProperty}, \texttt{?p})}, (\texttt{?u}, \texttt{?p}, \texttt{?u})\,.$$

(Note that this rule is valid with respect to OWL 2/RDF Based Semantics, but is not included in OWL 2 RL(/RDF).) Here, we see that `?u` *appears twice in the A-body of the rule. Now, take the following terminological axioms:*

1. (_:narcissist, owl:hasSelf, true)
2. (_:narcissist, owl:onProperty, ex:loves)
3. (_:narcissist, rdfs:subClassOf, _:egomaniac)
4. (_:egomaniac, owl:maxCardinality, 1)
5. (_:egomaniac, owl:onProperty, ex:loves)

and the assertional axioms:

6. (ex:freddy, ex:loves, ex:Fredrick)
7. (ex:freddy, ex:loves, ex:fred)
8. (ex:freddy, owl:sameAs, ex:Fredrick)

where by eq-rep-o *we can infer:*

9. (ex:freddy, ex:loves, ex:freddy) .

Now, by triples (1), (2), *and* (9), *and rule* prp-hs2, *we* should *infer:*

10. (ex:freddy, a, _:narcissist)

but this would require prp-hs2 *to be applied* after *consolidation. Thus, in a similar manner to Example 7.2, we would subsequently miss:*

11. (ex:freddy, a, _:egomaniac)
12. (ex:fred, owl:sameAs, ex:Fredrick)
13. (ex:Fredrick, owl:sameAs, ex:fred)

By rerunning our consolidation process a second time, we would infer triple (10)*—from triple* (9) *and rule* prp-hs2*—and then find the latter* `owl:sameAs` *relations.* ◊

Although we could consider recursively rerunning the consolidation process until fixpoint, we currently do not see this as worthwhile since: (i) we do not want `owl:sameAs` to affect the terminology of the data; (ii) our rules do not have "assertional join-variables" as per the latter example. Acknowledging the possibility of indeliberate incompleteness, we move on to discuss our distributed implementation.

7.4.2 Distributed approach

The distributed approach follows quite naturally from the previous discussion. As before, we assume that the input data are evenly pre-distributed over the slave machines (in any arbitrary ordering), where we can then apply the following process:

1. **run**: scan the distributed corpus (split over the slave machines) in parallel to extract relevant terminological knowledge;
2. **gather**: gather terminological data onto the master machine and thereafter ground the terminological atoms of the general/consolidation rules;
3. **flood**: flood the rules for reasoning and the consolidation-relevant patterns to all slave machines;
4. **run**: apply reasoning and extract consolidation-relevant statements from the input and inferred data;
5. **gather**: gather all consolidation statements onto the master machine, then in parallel:
    - **local**: compute the closure of the consolidation rules and the `owl:sameAs` data on the master machine;
    - **run**: each slave machine sorts its fragment of the main corpus according to natural order (s, p, o, c);
6. **flood**: send the closed `owl:sameAs` data to the slave machines once the distributed sort has been completed;
7. **run**: each slave machine then rewrites the subjects of their segment of the corpus, subsequently sorts the rewritten data by object, and then rewrites the (non-literal) objects (of non-`rdf:type` triples) with respect to the closed `owl:sameAs` data.

7.4.3 Performance Evaluation

Applying the above process to our 1.118 billion quadruple corpus took 12.34 h: we note that this is significantly more time-consuming than the baseline approach (1.06 h)—and even than the reasoning approach of Chapter 5 (3.35 h)—due to the large amount of on-disk sorts, scans and merge-joins required.

Extracting the terminological data took 1.14 h with an average idle time of 19 min (27.7%).Merging and aggregating the terminological data took roughly ∼1 min. Applying the reasoning and extracting the consolidation relevant statements took 2.34 h, with an average idle time of 2.5 min (1.8%). Aggregating and merging the consolidation relevant statements took 29.9 min. Thereafter, locally computing the closure of the consolidation rules and the equality data took 3.52 h, with the computation requiring two iterations overall (the minimum possible—the second iteration did not produce any new results); concurrent to the previous step, the parallel sort of remote data by natural order took 2.33 h with an average idle time of 6 min (4.3%). Subsequent parallel consolidation of the data took 4.8 h with 10 min (3.5%) average idle time—of this, ∼19% of the time was spent consolidating the pre-sorted subjects, ∼60% of the time was spent sorting the rewritten data by object, and ∼21% of the time was spent consolidating the objects of the data.

As before, Table 7.4 summarises the timing of the task, where the master machine requires 4.06 h to coordinate global knowledge, constituting the lower bound on time possible for the task to execute with respect to increasing machines in our setup—in future it may be worthwhile to investigate distributed strategies for computing the `owl:sameAs` closure (which takes 28.5% of the total computation time), but for the moment we mitigate the cost by concurrently running a sort on the slave machines, thus keeping the slaves busy for 63.4% of the time taken for this local aggregation step.[18] The slave machines were, on average, busy for 80.9% of the total task time; of the idle time, 73.3% was spent waiting for the master machine to aggregate the consolidation relevant data and to finish the closure of `owl:sameAs` data, and the balance (26.7%) was spent waiting for peers to finish (mostly during the extraction of terminological data).

Briefly, we also ran the consolidation without the general reasoning rules (Table B.4) motivated earlier.

[18] In future, parallelising the underlying sort operations may be sufficient to greatly enhance efficiency.

Category	**min**	**% Total**
Total execution time	740.4	100
Master (Local)		
Executing	243.6	32.9
Aggregate Consolidation Relevant Data	29.9	4
Closing `owl:sameAs`	*211.2*	*28.5*
Miscellaneous	2.5	0.3
Idle (waiting for slaves)	496.8	67.1
Slave (Parallel)		
Avg. Executing (total)	599.1	80.9
Extract Terminology	49.4	6.7
Extract Consolidation Relevant Data	137.9	18.6
Initial Sort (by subject)	*133.8*	*18.1*
Consolidation	278	37.5
Avg. Idle	141.3	19.1
Waiting for peers	37.5	5.1
Waiting for master	103.8	14

Table 7.4: Breakdown of timing of distributed extended consolidation with reasoning, where the two italicised tasks run concurrently on the master *and* slaves

With respect to performance, the main variations were given by (i) the extraction of consolidation relevant statements—this time directly extracted from explicit statements as opposed to explicit and inferred statements—which took 15.4 min (11% of the time taken including the general reasoning) with an average idle time of less than one minute (6% average idle time); (ii) local aggregation of the consolidation relevant statements took 17 min (56.9% of the time taken previously); (iii) local closure of the `owl:sameAs` data took 3.18 h (90.4% of the time taken previously). The total time saved equated to 2.8 h (22.7%), where 33.3 min were saved from coordination on the master machine, and 2.25 h were saved from parallel execution on the slave machines.

7.4.4 Results Evaluation

Note that in this section, we present the results of the consolidation which included the general reasoning step in the extraction of consolidation-relevant statements. In fact, we found that the only major variation between the two approaches was in the amount of consolidation-relevant statements collected (discussed presently), where other variations were in fact negligible (<0.1%). Thus, for our corpus, extracting only asserted consolidation-relevant statements offered a very close approximation of the extended reasoning approach.[19]

Extracting the terminological data, we found authoritative declarations of 434 functional properties, 57 inverse-functional properties, and 109 cardinality restrictions with a value of `1`.

As per the baseline consolidation approach, we again gathered 11.93 million `owl:sameAs` statements, as well as 52.93 million memberships of inverse-functional properties, 11.09 million memberships of functional properties, and 2.56 million cardinality-of-one relevant triples. Of these, respectively 22.14 million (41.8%), 1.17 million (10.6%) and 533 thousand (20.8%) were asserted—however, in the resulting closed `owl:sameAs` data derived with and without the extra reasoned triples, we detected a variation of less than 12 thousand terms (0.08%), where only 129 were URIs, and where other variations in statistics were less than 0.1% (e.g.,

[19] At least in terms of pure *quantity*. However, we do not give an indication of the *quality* or *importance* of those few equivalences we miss with this approximation, which may be application specific.

#	**Blacklisted Term**	**Occurrences**
1	empty literals	584,735
2	`<http://null>`	414,088
3	`<http://www.vox.com/gone/>`	150,402
4	`"08445a31a78661b5c746feff39a9db6e4e2cc5cf"`	58,338
5	`<http://www.facebook.com>`	6,988
6	`<http://facebook.com>`	5,462
7	`<http://www.google.com>`	2,234
8	`<http://www.facebook.com/>`	1,254
9	`<http://google.com>`	1,108
10	`<http://null.com>`	542

Table 7.5: Top ten most frequently occurring blacklisted values

there were 67 less equivalence classes when the reasoned triples were included).

From previous experience [Hogan et al., 2007a], we were aware of certain values for inverse-functional properties and functional properties which are erroneously published by exporters and which cause massive incorrect consolidation. We thus blacklist statements featuring such values from our consolidation processing, where we give the top 10 such values encountered for our corpus in Table 7.5—this blacklist is the result of trial and error, manually inspecting large equivalence classes and the most common values for (inverse-)functional properties. Empty literals are commonly exported (with and without language tags) as values for inverse-functional-properties (particularly FOAF "chat-ID properties"). The literal `"08445a31a78661b5c746feff39a9db6e4e2cc5cf"` is the SHA-1 hash of the string '`mailto:`', commonly assigned as a `foaf:mbox_sha1sum` value to users who don't specify their email in some input form. The remaining URIs are mainly user-specified values for `foaf:homepage`, or values automatically assigned for users that don't specify such.[20]

During the computation of the `owl:sameAs` closure, we found zero inferences through cardinality rules, 106.8 thousand raw `owl:sameAs` inferences through function-property reasoning, and 8.7 million raw `owl:sameAs` inferences through inverse-functional-property reasoning. The final canonicalised, closed, and non-symmetric `owl:sameAs` index (such that s_1 `SA` s_2, $s_1 > s_2$, and s_2 is a canon) contained 12.03 million statements.

From this data, we generated 2.82 million equivalence classes (an increase of 1.31× from baseline consolidation) mentioning a total of 14.86 million terms (an increase of 2.58× from baseline—5.77% of all URIs and blank-nodes), of which 9.03 million were blank-nodes (an increase of 2173× from baseline—5.46% of all blank-nodes) and 5.83 million were URIs (an increase of 1.014× from baseline—6.33% of all URIs). Thus, we see a large expansion in the amount of blank-nodes consolidated, but only minimal expansion in the set of URIs referenced in the equivalence classes. With respect to the canonical identifiers, 641 thousand (22.7%) were blank-nodes and 2.18 million (77.3%) were URIs.

Figure 7.5 contrasts the equivalence class sizes for the baseline approach (seen previously in Figure 7.3), and for the extended reasoning approach. Overall, there is an observable increase in equivalence class sizes, where we see the average equivalence class size grow to 5.26 entities (1.98× baseline), the largest equivalence class size grow to 33,052 (3.9× baseline) and the percentage of equivalence classes with the minimum size 2 drop to 63.1% (from 74.1% in baseline).

In Table 7.6, we update the five largest equivalence classes. Result 2 carries over from the baseline consolidation. The rest of the results are largely intra-PLD equivalences, where the entity is described using thousands of blank-nodes, with a consistent (inverse-)functional property value attached. Result 1 refers to a

[20] Our full blacklist contains forty-one such values, and can be found at `http://aidanhogan.com/swse/blacklist.txt`.

#	Canonical Term (Lexically Lowest in Equivalence Class)	Size	OK?
1	bnode37@http://a12iggymom.vox.com/profile/foaf.rdf	33,052	✓
2	http://bio2rdf.org/dailymed_drugs:1000	8,481	✗
3	http://ajft.org/rdf/foaf.rdf#_me	8,140	✓
4	bnode4@http://174.129.12.140:8080/tcm/data/association/100	4,395	✓
5	bnode1@http://aaa977.vox.com/profile/foaf.rdf	1,977	✓

Table 7.6: Largest 5 equivalence classes after extended consolidation

meta-user—labelled `Team Vox`—commonly appearing in user-FOAF exports on the Vox blogging platform.[21] Result 3 refers to a person identified using blank-nodes (and once by URI) in thousands of RDF documents resident on the same server. Result 4 refers to the Image Bioinformatics Research Group in the University of Oxford—labelled `IBRG`—where again it is identified in thousands of documents using different blank-nodes, but a consistent `foaf:homepage`. Result 5 is similar to Result 1, but for a Japanese version of the Vox user.

Figure 7.6 presents a similar analysis to Figure 7.5, this time looking at identifiers on a PLD-level granularity. Interestingly, the difference between the two approaches is not so pronounced, initially indicating that many of the additional equivalences found through the consolidation rules are "intra-PLD". In the baseline consolidation approach, we determined that 57% of equivalence classes were inter-PLD (contain identifiers from more that one PLD), with the plurality of equivalence classes containing identifiers from precisely two PLDs (951 thousand, 44.1%); this indicates that explicit `owl:sameAs` relations are commonly asserted between PLDs. In the extended consolidation approach (which of course subsumes the above results), we determined that the percentage of inter-PLD equivalence classes dropped *to* 43.6%, with the majority of equivalence classes containing identifiers from only one PLD (1.59 million, 56.4%). The entity with the most diverse identifiers (the observable outlier on the x-axis in Figure 7.6) was the person "Dan Brickley"—one of the founders and leading contributors of the FOAF project—with 138 identifiers (67 URIs and 71 blank-nodes) minted in 47 PLDs; various other prominent community members and some country identifiers also featured high on the list.

In Table 7.7, we compare the consolidation of the top five ranked identifiers in the SWSE system (see [Hogan et al., 2010b]). The results refer respectively to (1) the (co-)founder of the Web "Tim Berners-Lee"; (2) "Dan Brickley" as aforementioned; (3) a meta-user for the micro-blogging platform StatusNet which exports RDF; (4) the "FOAF-a-matic" FOAF profile generator (linked from many diverse domains hosting FOAF profiles it created); and (5) "Evan Prodromou", founder of the `identi.ca`/StatusNet micro-blogging service and platform. We see a significant increase in equivalent identifiers found for the first two results; however, we also noted that after reasoning consolidation, Dan Brickley was conflated with a second person.[22]

Note that the most frequently co-occurring PLDs in our equivalence classes remained unchanged from Table 7.3.

During the rewrite of the main corpus, terms in 151.77 million subject positions (13.58% of all subjects) and 32.16 million object positions (3.53% of non-`rdf:type` objects) were rewritten, giving a total of 183.93 million positions rewritten (1.8× the baseline consolidation approach). In Figure 7.7, we compare the reuse of terms across PLDs before consolidation, after baseline consolidation, and after the extended reasoning consolidation. Again, although there is an increase in reuse of identifiers across PLDs, we note that: (i) the vast majority of identifiers (about 99%) still only appear in one PLD; (ii) the difference between the baseline

[21] This site shut down on 2010/09/30.

[22] Domenico Gendarmi with three URIs—one document assigns one of Dan's `foaf:mbox_sha1sum` values (for `danbri@w3.org`) to Domenico: `http://foafbuilder.qdos.com/people/myriamleggieri.wordpress.com/foaf.rdf`; retr. 2010/11/27.

#	Canonical Term	BL#	R#
1	`<http://www4.wiwiss.fu-berlin.de/dblp/resource/person/Tim_Berners-Lee>`	26	50
2	`<genid:danbri>`	10	138
3	`<http://update.status.net/>`	0	0
4	`<http://www.ldodds.com/foaf/foaf-a-matic>`	0	0
5	`<http://update.status.net/user/1#acct>`	0	6

Table 7.7: Equivalence class sizes for top five SWSE-ranked identifiers with respect to baseline (BL#) and reasoning (R#) consolidation

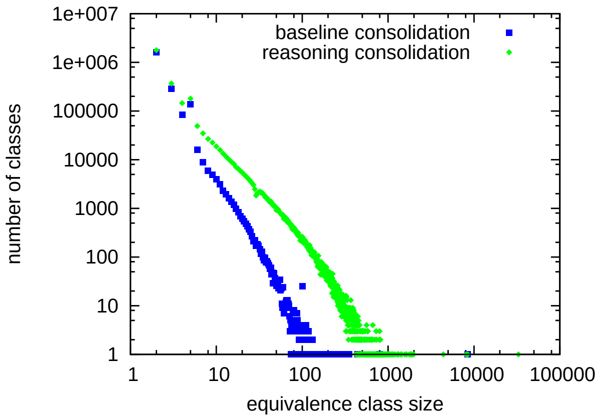


Figure 7.5: Distribution of the number of identifiers per equivalence classes for baseline consolidation and extended reasoning consolidation [log/log]

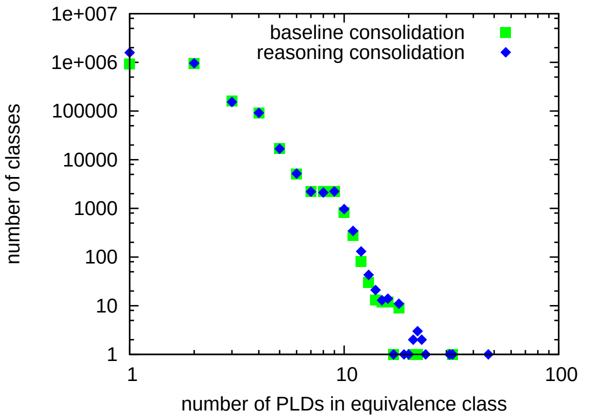


Figure 7.6: Distribution of the number of PLDs per equivalence class for baseline consolidation and extended reasoning consolidation [log/log]

and extended reasoning approach is not so pronounced. The most widely referenced consolidated entity—in terms of unique PLDs—was "Evan Prodromou" as aformentioned, referenced with six equivalent URIs in 101 distinct PLDs.

In summary, we see that applying the consolidation rules with respect to only asserted data—i.e., not considering the inferences given by $\mathcal{O}2\mathcal{R}^{-}$—is a good approximation for our Linked Data corpus, and that in comparison to the baseline consolidation over explicit `owl:sameAs`, (i) the additional consolidation rules offer a large bulk of intra-PLD consolidation of blank-nodes with large equivalence-class sizes, which we believe to be due to publishing practices whereby a given exporter uses consistent inverse-functional property values instead of URIs to uniquely identify entities across local documents; and (ii) where there is only a minor expansion (1.014×) in the number of URIs involved in the consolidation.

7.5 Statistical Concurrence Analysis (Synopsis)

In Appendix D, we present an approach for deriving a statistical measure which quantifies a form of similarity between entities based on the number (and selectivity) of inlinks and outlinks they share (including literal-valued attributes); we call this measure *concurrence*. In fact, we initially investigated this approach as a means of deriving coreference (and associated confidence values) based on statistical measures, but where the results were inconclusive [Hogan et al., 2010d].[23] However, we do see the approach as a useful similarity

[23] From the results presented in § D.4, clearly those entity pairs with the highest concurrence value are *not* coreferent.

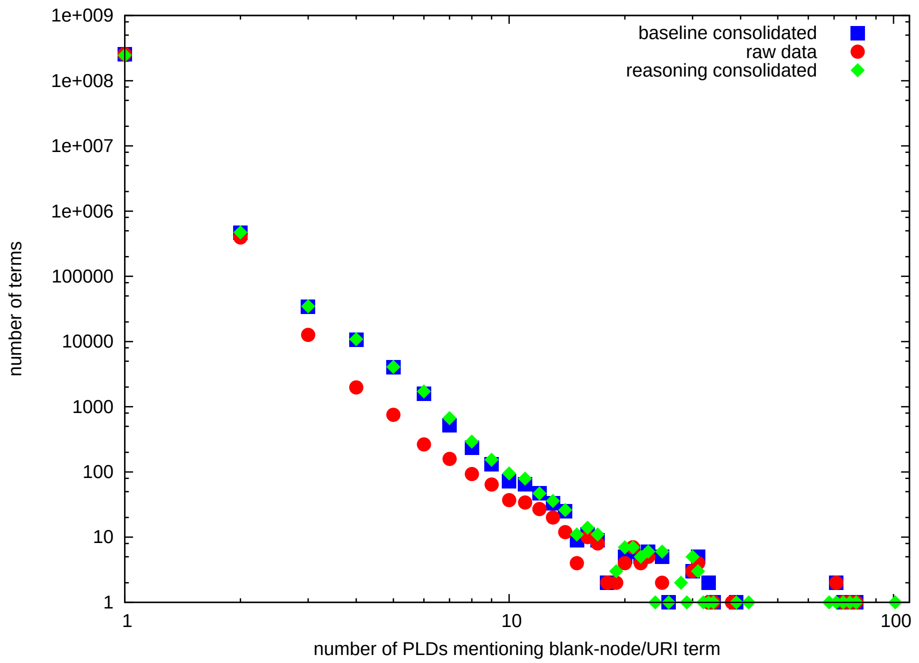


Figure 7.7: Distribution of number of PLDs terms are referenced by, for the raw, baseline consolidated, and reasoning consolidated data (log/log)

measure for entities in Linked Data, which can be derived in a scalable, distributed and domain-agnostic manner. In the following section, we use concurrence values for repairing unsatisifiable equivalence classes (equivalence classes which provoke inconsistency), and refer the interested reader to Appendix D for more detail on how these measures are computed.[24] However, for the purposes of the next chapter, it is sufficient to intuitively consider concurrence as a similarity measure which is computed based on the nature of the edges shared between two entities.

7.6 Entity Disambiguation

We have already seen that—even by only exploiting the formal logical consequences of the data through reasoning—consolidation may already be imprecise. Herein, we sketch and demonstrate an initial approach to identify 'incorrect' consolidation by means of inconsistency analysis, and a subsequent repair strategy based on statistical concurrence scores previously outlined. *We note at the outset that evaluation of this preliminary disambiguation approach against our corpus only identified a small amount of incorrect coreference.*

7.6.1 High-level Approach

The high-level approach is to see if the consolidation of any entities conducted in § 7.4.4 lead to any *novel* inconsistencies, and subsequently *recant* the equivalences involved; thus, it is important to note that our aim is not to repair inconsistencies in the data—as presented in Chapter 6—but instead to repair incorrect consolidation symptomised by inconsistency; further, we take a different approach to that of Chapter 6 since we now deal with rules involving more than one A-atom, and we wish to offer more specific optimisations

[24] We provide this (somewhat lengthy) discussion in the appendix—and not directly herein—so as to avoid breaking the logical flow of the chapter: baseline consolidation → extended consolidation → disambiguation.

in-tune with our consolidation approach.

Moving forward, herein we aim to (i) describe what forms of inconsistency we detect; (ii) characterise how inconsistencies can be caused by consolidation using examples from our corpus where possible; and (iii) sketch our proposal for repairing equivalence classes which have been determined to cause inconsistency.

First, recall from § 7.4.4 that we output sextuples of the form:

$$(s, p, o, c, s', o')$$

where (s, p, o, c) denote the consolidated quadruple containing canonical identifiers in the subject/object position as appropriate, and where s' and o' track the input identifiers prior to consolidation.

To detect inconsistencies in the consolidated corpus, we again use the OWL 2 RL/RDF rules with the `false` consequent [Grau et al., 2009] as listed in Table B.6, and which we again call constraints (see § 6.3.7). We italicise the labels of rules requiring new OWL 2 constructs, where we expect few such axioms to appear on the Web: as per Table 5.2, one document provides 8 `owl:AsymmetricProperty` and 10 `owl:IrreflexiveProperty` axioms[25], and one directory gives 9 `owl:AllDisjointClasses` axioms[26], and where we found no other OWL 2 axioms relevant to the rules in Table B.6. In any case, we include all constraints with the exception this time of cls-maxqc1, eq-diff2, eq-diff3, prp-npa1 and prp-npa2, where as we will see in § 7.6.2, these constraints are incompatible with our current implementation which requires a consistent assertional join variable for computing a merge-join operation. Note that we found no `owl:maxQualifiedCardinality` axioms for cls-maxqc1 in our corpus (cf. Table 5.2), no negative property assertion axioms for prp-npa*, although we did find 68 `owl:AllDifferentFrom` axioms for eq-diff*.

We also consider an additional constraint, whose semantics are indirectly axiomatised by the OWL 2 RL/RDF rules (through prp-fp, dt-diff and eq-diff1), but which we must support directly since we do not consider consolidation of literals:

$$\leftarrow \underline{(\texttt{?p, a, owl:FunctionalProperty})}, (\texttt{?x, ?p, ?l}_1),(\texttt{?x, ?p, ?l}_2),(\texttt{?l}_1, \texttt{owl:differentFrom}, \texttt{?l}_2)$$
$$[\texttt{?l}_1,\texttt{?l}_2 \in \mathsf{L}]$$

where we underline the terminological atom; we denote this rule by prp-fp2. To illustrate, we present an example from our corpus in Listing 7.5. (Note that for the examples presented herein, we will use the original identifiers, but will underline those considered coreferent by consolidation; we also endeavour to use real examples from our corpus wherever possible.) Here we see two very closely related models of cars consolidated in the previous step, but where we now identify that they have two different values for `dbo:length`—a functional-property—and thus consolidation raises an inconsistency.

Listing 7.5: Inconsistent functional datatype values

```
# Terminological [http://dbpedia.org/data3/length.rdf]
 dbo:length rdf:type owl:FunctionalProperty .

# Assertional [http://dbpedia.org/data/Fiat_Nuova_500.xml]
 dbpedia:Fiat_Nuova_500 dbo:length "3.546"^^xsd:double .

# Assertional [http://dbpedia.org/data/Fiat_500.xml]
 dbpedia:Fiat_Nuova dbo:length "2.97"^^xsd:double .
```

Further, note that we do not expect `owl:differentFrom` assertions to be materialised, but instead intend

[25] `http://models.okkam.org/ENS-core-vocabulary#country_of_residence`; retr. 2010/11/27

[26] `http://ontologydesignpatterns.org/cp/owl/fsdas/`; retr. 2010/11/27

a rather more relaxed semantics based on a heurisitic comparison: given two (distinct) literals substituted for $\texttt{?l}_1$ and $\texttt{?l}_2$, we flag an inconsistency iff (i) the data values of the two literals are not equal (standard OWL semantics); *and* (ii) their lower-case lexical values (i.e., the literal strings without language-tags and datatypes) are not equal. In particular, this relaxation is inspired by the definition of the FOAF (datatype) functional properties foaf:age, foaf:gender, and foaf:birthday, where the range of these properties is rather loosely defined: a generic range of rdfs:Literal is formally defined for these properties, with informal recommendations to use male/female as gender values, and MM-DD syntax for birthdays, but not giving recommendations for datatype or language-tags—our relaxation means that we would not flag an inconsistency in the illustrative example presented in Listing 7.6, and that we conservatively underestimate such inconsistency which may be caused by simple, innocuous syntactic variation in literals across published documents.

Listing 7.6: Example which is consistent when using heuristic literal matching

```
# Terminological [http://xmlns.com/foaf/spec/index.rdf]
 foaf:Person owl:disjointWith foaf:Document .

# Assertional [fictional]
 ex:Ted foaf:age 25 .
 ex:Ted foaf:age "25" .
 ex:Ted foaf:gender "male" .
 ex:Ted foaf:gender "Male"@en .
 ex:Ted foaf:birthday "25-05"^^xsd:gMonthDay .
 ex:Ted foaf:birthday "25-05" .
```

With respect to the constraints, we assume the terminological data to be sound, but only consider authoritative terminological axioms.[27]

For each grounding of a constraint, we wish to analyse the join positions to determine whether or not the given inconsistency is caused by consolidation; we are thus only interested in join variables which appear at least once in a data-level position (§ 7.2—in the subject position or object position of a non-rdf:type triple) and where the join variable is "intra-assertional" (exists twice in the assertional atoms). Thus, we are not interested in the constraint cls-nothing:

$$\leftarrow (?x, \texttt{rdf:type}, \texttt{owl:Nothing})$$

since it cannot be caused *directly* by consolidation: any grounding of the body of this rule must also exist (in non-canonical form) in the input data. For similar reasons, we also omit the constraint dt-not-type which looks for ill-typed literals: such literals must be present prior to consolidation, and thus the inconsistency detected by this constraint is not *directly* caused by consolidation—we see such constraints as unsuitable for detecting/diagnosing problems with coreference (they echo a pre-existing condition).

Moving forward, first note that owl:sameAs atoms—particularly in rule eq-diff1—are implicit in the consolidated data; e.g., consider the example of Listing 7.7 where an inconsistency is implicitly given by the owl:sameAs relation that holds between the consolidated identifiers wikier:wikier and eswc2006p:sergio-fernandez. In this example, there are two Semantic Web researchers, respectively named "Sergio Fernández"[28] and "Sergio Fernández Anzuola"[29] who both participated in the ESWC 2006

[27] In any case, we always source terminological data from the raw unconsolidated corpus.

[28] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/f/Fern=aacute=ndez:Sergio.html; retr. 2010/11/27

[29] http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/a/Anzuola:Sergio_Fern=aacute=ndez.html; retr. 2010/11/27

conference, and who were subsequently conflated in the "DogFood" export.[30] The former Sergio subsequently added a counter-claim in his FOAF file, asserting the above `owl:differentFrom` statement.

Listing 7.7: Different-from assertion

```
# Assertional [http://www.wikier.org/foaf.rdf]
 wikier:wikier owl:differentFrom eswc2006p:sergio-fernandez .
```

Other inconsistencies do not involve explicit `owl:sameAs` atoms, a subset of which may require "positive" reasoning to be detected; we provide an example from our corpus in Listing 7.8 where we found the description of the W3C organisation to be inconsistent as follows: (i) two W3C resources are initially consolidated due to sharing the value `http://www.w3.org/` for the inverse-functional property `foaf:homepage`; (ii) the W3C is stated to be a `foaf:Organization` in one document, and is inferred to be a person from its `identi.ca` profile through rule prp-dom (due to having `foaf:knows` links attached); finally, (iii) the W3C is a member of two disjoint classes, forming an inconsistency detectable by rule cax-dw.[31]

Listing 7.8: The W3C is inconsistent

```
# Terminological [http://xmlns.com/foaf/spec]
 foaf:Person owl:disjointWith foaf:Organization .
 foaf:knows rdfs:domain foaf:Person .

# Assertional [http://identi.ca/w3c/foaf]
 identica:48404 foaf:knows identica:45563 .

# Assertional [inferred by prp-dom]
 identica:48404 a foaf:Person

# Assertional [http://data.semanticweb.org/organization/w3c/rdf]
 sworg:w3c a foaf:Organization .
```

In order to resolve such inconsistencies, we make three simplifying assumptions:

1. the steps involved in the consolidation can be rederived with knowledge of direct inlinks and outlinks of the consolidated entity, or reasoned knowledge derived therefrom;
2. inconsistencies are caused by pairs of consolidated identifiers;
3. we repair individual equivalence classes and do not consider the case where repairing one such class may indirectly repair another (i.e., we do not guarantee a "globally minimal" set of repairs, but only consider repair options for each individual equivalence class).

With respect to the first item, our current implementation performs a repair of the equivalence class based on knowledge of direct inlinks and outlinks, available through a simple merge-join as used in the previous section; this thus precludes repair of consolidation found through rules eq-diff2, eq-diff3, prp-npa1, prp-npa2 and cls-maxqc2, which also require knowledge about assertional triples not directly associated with the consolidated entity (cf. § 7.4.1)—for example, cls-maxqc2 also requires information about the class memberships of the resources linked to by the consolidated entity.

[30] `http://data.semanticweb.org/dumps/conferences/eswc-2006-complete.rdf`; retr. 2010/11/27

[31] Note that this also could be viewed as a counter-example for using inconsistencies to recant consolidation, where arguably the two entities are coreferent from a practical perspective, even if "incompatible" from a symbolic perspective.

With respect to the second item, we assume that inconsistencies are caused by pairs of identifiers, such that we only consider inconsistencies caused by what we call "unsatisfiable coreference" and do not consider the case where the alignment of more than two identifiers are required to cause a single inconsistency (not possible in our rules) where such a case would again lead to a disjunction of repair strategies.

With respect to the third item, it is possible to resolve a set of inconsistent equivalence classes by repairing one; for example, consider rules with multiple "intra-assertional" join-variables (prp-irp, prp-asyp) which can have explanations involving multiple consolidated identifiers, as demonstrated in the example of Listing 7.9 where both equivalences together—(`ex:A`, `owl:sameAs`, `ex:a`), (`ex:B`, `owl:sameAs`, `ex:b`)—constitute an inconsistency. Repairing one equivalence class would repair the inconsistency detected for both: we give no special treatment to such a case, and resolve each equivalence class independently, In any case, we find no such incidences in our corpus: these inconsistencies require (i) axioms new in OWL 2 (rules prp-irp, prp-asyp, prp-pdw and prp-adp); (ii) alignment of two consolidated sets of identifiers in the subject/object positions. Note that such cases can also occur given the recursive nature of our consolidation—consolidating one set of identifiers may lead to alignments in the join positions of the consolidation rules in the next iteration—however, we did not encounter such recursion during the consolidation phase (cf. 7.4.3). Thus, our third simplifying assumption (and its implications) has no bearing for our *current* corpus, where we observe that repairing one equivalence class cannot lead to the repair of another.

Listing 7.9: Example of an ambiguous inconsistency

```
# Terminological
 ex:made owl:propertyDisjointWith ex:maker .

# Assertional
 ex:A ex:maker ex:B .
 ex:a ex:made ex:b .
```

Thereafter, the high-level approach to repairing unsatisfiable coreference involves examining each consolidated entity—both its inlinks and outlinks—independently, looking for inconsistency, isolating the pairs of identifiers that cause said inconsistency, and thereafter repairing the equivalence class, revising the consolidation to reflect the repairs. For repairing the equivalence class, our approach is to deconstruct the equivalence class into a set of singletons, and thereafter begin to reconstruct a new set of equivalence classes from these singletons by iteratively merging the most "strongly linked" intermediary equivalence classes which will not contain incompatible identifiers: i.e., the equivalence classes for which the strongest evidence for coreference exists between its members. In more detail, the process of repairing each equivalence class is as follows:

1. use the constraints to discover pairs of identifiers which together cause inconsistency and must be separated;

2. assign each identifier in the original equivalence class into a consistent singleton equivalence class;

3. starting with the singletons, iteratively merge consistent equivalence classes, which do not together contain a pair of incompatible identifiers, and between which the *strongest evidence* for coreference exists, based on:

   (a) the number of different proofs (sets of input triples which infer) coreference between the equivalence classes;

   (b) if tied, use a concurrence score between the new identifier and the (merged) equivalence class (cf. Appendix D).

Following these intuitions, we can *sketch* a formalism of the repair thus: we denote the graph of non-transitive equivalences for a given equivalence class as a weighted graph $G = (V, E, \omega)$ such that $V \subset \mathsf{B} \cup \mathsf{U}$ is the set of vertices, $E \subset \mathsf{B} \cup \mathsf{U} \times \mathsf{B} \cup \mathsf{U}$ is the set of edges, and $\omega : E \mapsto \mathbb{N} \times \mathbb{R}$ is a weighting function for the edges. Our edge weights are pairs (d, c) where d is the number of sets of input triples in the corpus which allow to directly derive the given equivalence relation by means of a direct `owl:sameAs` assertion (in either direction), or a shared inverse-functional object, or functional subject—loosely, the independent evidences for the relation given by the input graph, excluding transitive `owl:sameAs` semantics; c is the concurrence score derivable between the unconsolidated entities and is used to resolve ties (we would expect many strongly connected equivalence graphs where, e.g., the entire equivalence class is given by a single shared value for a given inverse-functional property, and thus require the additional granularity of concurrence for repairing the data in a non-trivial manner). We define a total lexicographical order over these pairs.

Given an equivalence class $Eq \subset \mathsf{U} \cup \mathsf{B}$ which we perceive to cause a *novel* inconsistency—i.e., an inconsistency derivable by the alignment of incompatible identifiers—by application of the constraints over the inlinks and outlinks of the consolidated entity, we first derive a collection of sets $\mathcal{C} = \{C_1, \ldots, C_n\}$, $\mathcal{C} \subset 2^{\mathsf{U} \cup \mathsf{B}}$, such that $\forall C_i \in \mathcal{C}$, $|C_i| = 2, C_i \subseteq Eq$, and where each C_i contains two incompatible identifiers. Note that $\mathcal{C}$ encodes the pairs of identifiers which cannot appear together in the repaired equivalence class: those elements of Eq not involved in inconsistency will not be contained within $\mathcal{C}$.

We then apply a simple *consistent clustering* of the equivalence class, loosely following the notions of a minimal cutting (see, e.g., [Stoer and Wagner, 1997]). For Eq, we create an initial set of singleton sets $\mathcal{E}_0$, each containing an individual identifier in the equivalence class (a partition).

Now let $\Omega(E_i, E_j)$ denote the aggregated weight of the edge considering the merge of the nodes of E_i and the nodes of E_j in the graph: the pair (d, c) such that d denotes the number of unique evidences for equivalence relations between all nodes in E_i and all nodes in E_j and such that c denotes the concurrence score considering the merge of entities in E_i and E_j—intuitively, the same weight as before, but applied as if the identifiers in E_i and E_j were consolidated in the graph. We can apply the following clustering:

- for each pair of sets $E_i, E_j \in \mathcal{E}_n$ such that $\nexists\{a, b\} \in \mathcal{C} : a \in E_i, b \in E_j$ (i.e., consistently mergeable subsets) identify the weights of $\Omega(E_i, E_j)$ and order the pairings;
- in descending (lexicographical) order with respect to the above weights, merge E_i, E_j pairs—such that neither E_i or E_j have already been merged in this iteration—producing $\mathcal{E}_{n+1}$ at iteration's end;
- iterate over n until fixpoint: i.e., until no more classes in $\mathcal{E}_n$ can be consistently merged.

The result of this process is a set of equivalence classes $\mathcal{E}$—a partition of the original Eq—such that no element of $\mathcal{E}$ contains incompatible identifiers. We can subsequently revise the consolidated data to reflect $\mathcal{E}$.

7.6.2 Implementing Disambiguation

The implementation of the above disambiguation process can be viewed on two levels: the *macro* level which identifies and collates the information about individual equivalence classes and their respectively consolidated inlinks/outlinks, and the *micro* level which repairs individual equivalence classes.

On the macro level, the task assumes input data sorted by both subject (s, p, o, c, s', o') and object (o, p, s, c, o', s'), again such that s, o represent canonical identifiers and s', o' represent the original identifiers (as per § 7.4.1). Note that we also require the asserted `owl:sameAs` relations encoded likewise. Given that all of the required information about the equivalence classes (their inlinks, outlinks, derivable equivalences and original identifiers) are gathered under the canonical identifiers, we can apply a straightforward merge-join on *s*-*o* over the sorted stream of data, batching together the data (inlinks, outlinks and original identifiers) for each consolidated entity.

On a micro level, we buffer each individual consolidated segment into an in-memory index; currently, these segments fit in memory, where for the largest equivalence classes we note that inlinks/outlinks are commonly duplicated—if this were not the case, one could consider using an on-disk index which should be feasible given that only small batches of the corpus are under analysis at each given time. We additionally require access to the relevant terminological knowledge required for reasoning, and the predicate-level statistics derived during from the concurrence analysis. We apply scan-reasoning and inconsistency detection over each batch, and for efficiency, skip over batches which do not contain incompatible identifiers.

For equivalence classes containing incompatible identifiers, we first determine the full set of such pairs through application of the inconsistency detection rules: usually, each detection gives a single pair, where we ignore pairs containing the same identifier (i.e., detections which would equally apply over the unconsolidated data). We check the pairs for a trivial solution: if all identifiers in the equivalence class appear in some pair, we check whether (i) no pair of identifiers can be consistently merged, in which case, the equivalence class must necessarily be completely disbanded; or (ii) one identifier appears in all pairs of incompatible identifiers in the equivalence class, and is incompatible with all other identifiers, in which case this problematic identifier can be removed from the equivalence class to derive the repair.

For non-trivial repairs, we begin with the set of singletons and then apply the iterations described in the previous section, where at the beginning of each iteration, we derive the evidences for equivalence between all remaining pairs of sets in the partition which can be consistently merged—based on explicit `owl:sameAs` relations, and those inferable from the consolidation rules—and merge the pairs of sets accordingly. In the case of a tie, we perform the concurrence analysis, which derives a form of similarity.

In the final step, we encode (only) the repaired equivalence classes in memory, and perform a final scan of the corpus (in natural sorted order), revising identifiers according to their repaired canonical term.

7.6.3 Distributed Implementation

Distribution of the task becomes straightforward, assuming that the slave machines have knowledge of terminological data, predicate-level statistics, and already have the consolidation encoding sextuples sorted and coordinated by hash on s and o. (Note that all of these data are present on the slave machines from previous tasks.)

Thus, we are left with two steps:

- **run**: each slave machine performs the above process on it's segment of the corpus, applying a merge-join over the data sorted by $(s, p, o, c, s'o')$ and (o, p, s, c, o', s') to derive batches of consolidated data, which are subsequently analysed, diagnosed, and a repair derived in memory;
- **gather/run**: the master machine gathers all repair information from all slave machines, and floods the merged repairs to the slave machines; the slave machines subsequently perform the final repair of the corpus.

7.6.4 Performance Evaluation

The total time taken for inconsistency-based disambiguation was 3.91 h. The inconsistency and equivalence class repair analysis took 2.87 h, with a significant average idle time of 24.4 min (14.16%): in particular, certain large batches of consolidated data took significant amounts of time to process, particularly to reason

Category	**min**	**% Total**
Total execution time	234.8	100
Master (Local)		
Executing	1	0.4
Miscellaneous	1	0.4
Idle (waiting for slaves)	99.6	99.6
Slave (Parallel)		
Avg. Executing (total exc. idle)	205.5	87.5
Identify inconsistencies and repairs	147.8	62.9
Repair Corpus	57.7	24.6
Avg. Idle	29.3	12.5
Waiting for peers	28.3	12.1
Waiting for master	1	0.4

Table 7.8: Breakdown of timing of distributed disambiguation and repair

over.[32] Subsequently repairing the corpus took 1.03 h, with an average idle time of 3.9 min.

In Table 7.8, we again summarise the timing of the task. Note that the aggregation of the repair information took a negligible amount of time, and where only a total of one minute is spent on the slave machine. Most notably, load-balancing is somewhat of an issue, causing slave machines to be idle for, on average, 12.5% of the total task time, mostly waiting for peers. This percentage—and the general load-balancing characteristic—would likely increase further, given more machines, or a higher scale of data.

7.6.5 Results Evaluation

As alluded to at the outset of this section, our discussion of inconsistency repair has been somewhat academic: from the total of 2.82 million consolidated batches to check, we found 523 equivalence classes (0.019%) causing novel inconsistency. Of these, 23 were detected through `owl:differentFrom` assertions, 94 were detected through distinct literal values for inverse-functional properties, and 406 were detected through disjoint-class constraints. We list the top five functional-properties given non-distinct literal values in Table 7.9 and the top five disjoint classes in Table 7.10—note that the `dbo:` functional-properties gave identical detections, and that the class `foaf:Person` is a subclass of `foaf:Agent`, and thus an identical detection is given twice.[33] All equivalence classes were broken into two repaired sub-equivalence class—furthermore, all had a trivial repair given by separating a single identifier appearing in each incompatible pair (with all original identifiers appearing in some pair). Thus, for the moment, our repair strategy is purely academic.[34]

7.7 Related Work

Work relating to entity consolidation has been researched in the area of databases for a number of years, aiming to identify and process co-referent signifiers, with works under the titles of record linkage, record fusion, merge-purge, instance fusion, and duplicate identification, and (ironically) a plethora of variations

[32] The additional expense is due to the relaxation of duplicate detection: we cannot consider duplicates on a triple level, but must consider uniqueness based on the entire sextuple to derive the information required for repair. Thus, we must apply many duplicate inferencing steps.

[33] Further, note again that between the time of the crawl and the time of writing, the FOAF vocabulary has removed disjointness constraints between the `foaf:Document` and `foaf:Person`/`foaf:Agent` classes.

[34] We also considered a dual form of the concurrence to detect incorrect equivalence classes: for example, to use the quasi-functional nature of `foaf:name` to repair consolidated entities with multiple such values. However, we noted in preliminary results that such analysis gave poor results for our corpus, where we noticed, for example, that (indeed, highly ranked) persons with multiple `foaf:weblog` values—itself measured to be a quasi-functional property—would be identified as incorrect.

#	Functional Property	Detections
1	foaf:gender	56
2	foaf:age	32
3	dbo:height/dbo:height/dbo:wheelbase/dbo:width	4
4	atomowl:body	1
5	loc:address	1

Table 7.9: Breakdown of inconsistency detections for functional-properties, where dbo: properties gave identical detections

#	Disjoint Class 1	Disjoint Class 2	Detections
1	foaf:Document	foaf:Person(/foaf:Agent)	312
2	ecs:Group	ecs:Individual	37
3	foaf:Organization	foaf:Person	24
4	foaf:Document	foaf:Agent	23
5	foaf:Person	foaf:Project	7

Table 7.10: Breakdown of inconsistency detections for disjoint-classes

thereupon; see [Newcombe et al., 1959; Michalowski et al., 2003; Chen et al., 2005; Bernstein et al., 2005], etc., and a survey by Elmagarmid et al. [2007]. Unlike our approach which leverages the declarative semantics of the data in order to be domain agnostic, such systems usually operate given closed schemas—similarly, they typically focus on string-similarity measures and statistical analysis. Haas et al. [2009] note that:

> "[...] *in relational systems where the data model does not provide primitives for making same-as assertions* [...] *there is a value-based notion of identity"*
>
> —**[Haas et al., 2009]**

However, we note that some works have focussed on leveraging semantics for such tasks in relation databases; e.g., Fan et al. [2009] leverage domain knowledge to match entities, where interestingly they state:

> *"Real life data is typically dirty...* [thus] *it is often necessary to hinge on the semantics of the data"*
>
> —**[Fan et al., 2009]**

Some other works—more related to Information Retrieval and Natural Language Processing—focus on extracting coreferent entity names from unstructured text in order to align the results of Named Entity Recognition where for example, Singh et al. [2010] present an approach to identify coreferences from a corpus of 3 million natural language "mentions" of persons, where they build compound "entities" out of the individual mentions.

With respect to RDF, one area of research also goes by the name *instance matching*: for example, in 2009, the Ontology Alignment Evaluation Initiative[35] introduced a new test track on instance matching.

Nikolov et al. [2008] present the KnoFuss architecture for aligning data on an assertional level; they identify a three phase process involving coreferencing (finding equivalent individuals), conflict detection (finding inconsistencies caused by the integration), and inconsistency resolution. For the coreferencing, the authors introduce and discuss approaches incorporating string similarity measures and class-based machine learning techniques. Although the high-level process is similar to our own, the authors do not address scalability concerns.

[35] OAEI. http://oaei.ontologymatching.org/; retr. 2010/11/27

Scharffe et al. [2009] identify four steps in aligning datasets: *align*, *interlink*, *fuse* and *post-process*. The align process identifies equivalences between entities in the two datasets, the interlink process materialises `owl:sameAs` relations between the two datasets, the aligning step merges the two datasets (on both a terminological and assertional level, possibly using domain-specific rules), and the *post-processing* phase subsequently checks the consistency of the output data. Although parts of this process echoes our own, they have yet to demonstrate large-scale evaluation, focussing on datasets containing 2.5 thousand entities.

Noessner et al. [2010] present an approach for aligning two A-Boxes described using the same T-Box; in particular they leverage similarity measures introduced by Stuckenschmidt [2009], and define an optimisation problem to identify the alignment which generates the most highest weighted similarity between the two A-Boxes under analysis: they use Integer Linear Programming to generate the optimal alignment, encoding linear constraints to enforce *valid* (i.e., consistency preserving), one-to-one, functional mappings. Although they give performance results, they do not directly address scalability. Their method for comparing entities is similar in practice to ours: they measure the "overlapping knowledge" between two entities, counting how many assertions are true about both. The goal is to match entities such that: (i) the resulting consolidation is consistent; and (ii) the measure of overlap is maximal.

Like us, Castano et al. [2008] approach instance matching from two distinct perspectives: (i) determine coreferent identifiers; and (ii) detect similar individuals based on the data they share. Much of their work is similar in principle to ours: in particular, they use reasoning for identifying equivalences and use a statistical approach for identifying properties "with high identification power". They do not consider use of inconsistency detection for disambiguating entities, and perhaps more critically, only evaluate with respect to a dataset containing ∼15 thousand entities.

With respect to URI naming on the Web, Bouquet et al. [2006] argue for a centralised naming architecture for minting URI signifiers for the Web; we see such a centralised "'naming authority" as going against the ad-hoc, decentralised, scale-free nature of the Web.

The Sindice and Sig.ma search systems internally use inverse-functional properties to find equivalent identifiers [Tummarello et al., 2007, 2009]. Sindice investigates some bespoke "schema-level" reasoning to identify a wider range of inverse-functional properties [Tummarello et al., 2007]; however, compared to our approach, they (i) do not use functional properties or cardinality constraints; (ii) would still miss equivalences where identifiers use the same value with different inverse-functional properties, and where, e.g., those properties are in an equivalence or subsumption relationship.

Online systems RKBExplorer [Glaser et al., 2008, 2009][36], <sameAs>[37] and ObjectCoref [Cheng and Qu, 2009][38] offer on-demand querying for `owl:sameAs` relations found for a given input URI, which they internally compute and store; the former focus on publishing `owl:sameAs` relations for authors and papers in the area of scientific publishing, with the latter two systems offering more general `owl:sameAs` relationships between Linked Data identifiers. In fact, many of the `owl:sameAs` relations we consume are published as Linked Data by the RKBExplorer system.

Volz et al. [2009] present the Silk framework for creating and maintaining inter-linkage between domain-specific RDF datasets; in particular, this framework provides publishers with a means of discovering and creating `owl:sameAs` links between data sources using domain-specific rules and parameters. Thereafter, publishers can integrate discovered links into their exports, enabling better linkage of the data and subsequent consolidation by data consumers: this framework goes hand-in-hand with our approach, producing the `owl:sameAs` relations which we consume.

[36] `http://www.rkbexplorer.com/sameAs/`; retr. 2011/01/22

[37] `http://sameas.org/`; retr. 2011/01/22

[38] `http://ws.nju.edu.cn/objectcoref/`; retr. 2011/01/22

Popitsch and Haslhofer [2010] present discussion on the problem of broken links in Linked Data, identifying structurally broken links (the Web of Data's version of a "deadlink") and semantically broken links, where the original meaning of an identifier changes after a link has been remotely asserted. The authors subsequently present the DSNotify system, which monitors dynamicity over a given subset of Linked Data and can detect and act upon changes—e.g., to notify another agent or correct broken links—and can also be used to indirectly link the dynamic target.

Various authors have looked at applying consolidation over domain-specific RDF corpora: e.g., Sleeman and Finin [2010] look at using machine learning techniques to consolidate FOAF personal profile information; Shi et al. [2008] similarly look at FOAF-specific alignment techniques using inverse-functional properties and fuzzy string matching; Jentzsch et al. [2009] examine alignment of published drug data;[39] Raimond et al. [2009] look at interlinking RDF from the music-domain; Monaghan and O'Sullivan [2007] apply consolidation to photo annotations expressed in RDF.

Salvadores et al. [2009] present the LinksB2N system which aims to perform scalable integration of RDF data, particularly focussing on evaluation over corpora from the marketing domain; however, their methods are not specific to this domain. They do not leverage the semantics of the data for performing consolidation, instead using similarity measures, based on the idea that "the unique combination of RDF predicates associated with RDF resources is what defines their existence as unique" [Salvadores et al., 2009]. This is a similar intuition to that behind our concurrence analysis, but we again question the validity of such an assumption for consolidation, particularly given incomplete data and the Open World Assumption underlying RDF(S)/OWL—we view an RDF resource as a description of something signified, and would wish to avoid conflating unique signifiers, even if they match *precisely* with respect to their description.

Halpin et al. [2010a] discuss the semantics and current usage of `owl:sameAs` in Linked Data, discussing issues relating to *identity*, and providing four categories of `owl:sameAs` usage to relate entities which are closely related, but for which the semantics of `owl:sameAs`—particularly substitution—does not quite hold; in fact, the discussion of Halpin et al. [2010a] serves as a counterpoint to those aforementioned related works which translate weighted similarities into weighted equivalences. Needless to say, we do not discount such approaches—they may of course of course derive useful and correct alignments not possible through a purely symbolic analysis—but we would be cautious when considering using such an approach over arbitrary Linked Data, particularly given the inherent difficulties in evaluating the precision thereof.

Cudré-Mauroux et al. [2009] present the idMesh system, which leverages user-defined associations and probabalistic methods to derive entity-level relationships, including resolution of conflicts; they also delineate entities based on "temporal discrimination", whereby coreferent entities may predate or postdate one another, capturing a description thereof at a particular point in time. The idMesh system itself is designed over a peer-to-peer network with centralised coordination. However, evaluation is over synthetic data, where they only demonstrate a maximum scale involving 8,000 entities and 24,000 links over 400 machines: the evaluation of performance focusses on network traffic and message exchange as opposed to time.

In the works of Kiryakov et al. [2009], Urbani et al. [2010] and Kolovski et al. [2010]—already discussed in Chapter 5—the authors incorporate rules dealing with equality in their reasoning engines, and also use optimisations similar to our canonicalisation as a necessary means of avoiding the quadratic nature of traditional replacement semantics for `owl:sameAs`.

[39] In fact, we believe that this work generates the incorrect results observable in Table 7.2; cf. `http://groups.google.com/group/pedantic-web/browse_thread/thread/ad740f7052cc3a2d` (retr. 2011/01/22).

7.8 Critical Discussion and Future Directions

In this section, we provide critical discussion of our approach, following the dimensions of the requirements listed at the outset.

With respect to **scale**, on a high level, our primary means of organising the bulk of the corpus is external-sorts, characterised by the linearithmic time complexity $O(n\log(n))$; external-sorts do not have a critical main-memory requirement, and are efficiently distributable. Our primary means of accessing the data is via linear scans. With respect to the individual tasks:

- our current baseline consolidation approach relies on an in-memory `owl:sameAs` index: however we demonstrate an on-disk variant in the extended consolidation approach;
- the extended consolidation currently loads terminological data into memory, which is required by all machines: if necessary, we claim that an on-disk terminological index would offer good performance given the distribution of class and property memberships, where we posit that a high cache-hit rate would be enjoyed;
- for the entity concurrency analysis, the predicate level statistics required by all machines are small in volume—for the moment, we do not see this as a serious factor in scaling-up;
- for the inconsistency detection, we identify the same potential issues with respect to terminological data; also, given large equivalence classes with a high number of inlinks and outlinks, we would encounter main-memory problems, where we posit that an on-disk index could be applied assuming a reasonable upper limit on batch sizes.

With respect to **efficiency**:

- the on-disk aggregation of `owl:sameAs` data for the extended consolidation has proven to be a bottleneck—for efficient processing at higher levels of scale, distribution of this task would we a priority, which should be feasible given that again, the primitive operations involved are external sorts and scans, with non-critical in-memory indices to accelerate reaching the fixpoint;
- although we typically observe terminological data to constitute a small percentage of Linked Data corpora (<0.1% in our corpus; cf. older results in [Hogan et al., 2009b, 2010c]) at higher scales, aggregating the terminological data for all machines may become a bottleneck, and distributed approaches to perform such would need to be investigated; similarly, as we have seen, large terminological documents can cause load-balancing issues;[40]
- for the concurrence analysis and inconsistency detection, data are distributed according to a modulo-hash function on the subject and object position, where we do not hash on the objects of `rdf:type` triples—although we demonstrated even data distribution by this approach for our current corpus, this may not hold in the general case;
- as we have already seen for our corpus and machine count, the complexity of repairing consolidated batches may become an issue given large equivalence class sizes;
- there is some notable idle time for our machines, where the total cost of running the pipeline could be reduced by interleaving jobs.

[40] We reduce terminological statements on a document-by-document basis according to unaligned blank-node positions: for example, we prune RDF collections identified by blank-nodes which do not join with, e.g., an `owl:unionOf` axiom.

With the exception of our manually derived blacklist for values of (inverse-)functional-properties, the methods presented herein have been entirely **domain-agnostic** and **fully automatic**.

One major open issue is the question of **precision** and **recall**. Given the nature of the tasks—particularly the scale and diversity of the datasets—we posit that deriving an appropriate gold standard is currently infeasible:

- the scale of the corpus precludes manual or semi-automatic processes;
- any automatic process for deriving the gold standard would make redundant the approach to test;
- results derived from application of the methods on subsets of manually verified data would not be equatable to the results derived from the whole corpus;
- even assuming a manual approach were feasible, oftentimes there is no objective criteria for determining what precisely signifies what—the publisher's original intent is often ambiguous.

Thus, we prefer symbolic approaches to consolidation and disambiguation which are predicated on the formal semantics of the data, where we can appeal to the fact that incorrect consolidation is due to erroneous data, not an erroneous approach. Without a formal means of sufficiently evaluating the results, we currently only employ statistical methods for applications where precision is not a primary requirement. In general, we posit that for the corpora we target, such research can only find it's real litmus test when integrated into a system with a critical user-base.

Finally, we have only briefly discussed issues relating to **Web-tolerance**: e.g., spamming or conflicting data. With respect to such consideration, we currently (i) derive and use a blacklist for common void values; (ii) consider authority for terminological data; and (iii) try to detect erroneous consolidation through consistency verification. With respect to (iii), an interesting research direction would be to investigate statistical approaches for identifying additional malignant coreference given by methods such as ours, in corpora such as ours. Further research into the benefits of different repair strategies for such coreference is also warranted: for example, empirical analysis may demonstrate that coreference given by direct `owl:sameAs` is, in the general case, more reliable than coreference given by inverse-functional properties—or perhaps vice-versa—which could lead to new repair strategies which define more trust in different types of coreference "proofs". Also, unlike our general repair of inconsistencies in Chapter 6, we have not considered leveraging the ranking scores of data-sources or triples in the repair; further investigation along these lines may also lead to more granular repair strategies.

Again broaching on the topic of **Web-tolerance**, our approach (naïvely) trusts all equivalences asserted or derived from the data until they are found to cause inconsistency: as such, we assume good faith on the part of the publishers, and deem all coreference as innocent until proven guilty—this is inarguably naïve for Web data, especially given that our current methods for diagnosing problematic coreference are quite coarse-grained. Acknowledging that our coreference is fallible, we track the original pre-consolidation identifiers—encoded in sextuples of the form (s, p, o, c, s', o')—which can be used by consumers to revert erroneous consolidation. In fact, similar considerations can be applied more generally to the reuse of identifiers across sources: giving special consideration to the consolidation of third party data about an entity is somewhat fallacious without also considering the third party contribution of data using a consistent identifier. In both cases, we track the context of (consolidated) statements which at least can be used to verify or post-process sources.[41] Currently, the corpus we evaluate our methods against does not exhibit any significant deliberate spamming, but rather indeliberate noise—we leave more mature means of handling spamming for future work (as required).

[41] Although it must be said, we currently do not track the steps used to derive the equivalences involved in consolidation, which would be expensive to materialise and maintain.

To wrap up this chapter, we have provided a comprehensive discussion on scalable and distributed methods for consolidating, matching, and disambiguating entities present in a large static Linked Data corpus. Throughout, we have focussed on the scalability and practicalities of applying our methods over real, arbitrary Linked Data in a domain agnostic and (almost entirely) automatic fashion. We have shown how to use explicit `owl:sameAs` relations in the data to perform consolidation, and subsequently expanded this approach, leveraging the declarative formal semantics of the corpus to materialise additional `owl:sameAs` relations. We also presented (albeit indirectly in Appendix D) a scalable approach to identify weighted entity concurrences: entities which share many inlinks, outlinks, and attribute values—we note that those entities demonstrating the highest concurrence were *not* coreferent. Next, we presented an approach using inconsistencies to disambiguate entities and subsequently repair equivalence classes: we found that this approach currently derives few diagnoses, where the granularity of inconsistencies within Linked Data is not sufficient for accurately pinpointing all incorrect consolidation. Finally, we tempered our contribution with critical discussion, particularly focussing on scalability and efficiency concerns.

We believe that consolidation and disambiguation—particularly as applied to large scale Linked Data corpora—is of particular significance given the rapid growth in popularity of Linked Data publishing. As the scale and diversity of the Web of Data expands, scalable and precise data integration technique will become of vital importance, particularly for data warehousing applications—we see the work presented herein as a significant step in the right direction.

Chapter 8

Discussion and Conclusion

> *"You have your way. I have my way. As for the right way, the correct way, and the only way, it does not exist."*
>
> **—Friedrich Nietzsche**

There has been a recent and encouraging growth in heterogeneous RDF documents published on the Web. Acting as a catalyst for this burgeoning adoption, the Linked Data community and the Linking Open Data project have advocated the tangible benefits of RDF and related Semantic Web technologies for publishing and interlinking open data on the Web in a standardised manner. The result is a novel Web of Data, which poses new challenges and research directions with respect to how this heterogeneous, unvetted and potentially massive corpus (or some interesting subset thereof) can be integrated in a manner propitious to subsequent consumers. Indeed, inherent heterogeneity poses significant obstacles with respect to how such data can be processed and queried, where scale and intrinsic noise preclude the applicability (and often the desirability) of standard reasoning techniques to smooth out this heterogeneity—this has been the motivating premise of this thesis.

Summary of Contributions

We now give a summary of the primary contributions of this thesis, as given by Chapters 4–7.

Crawling, Corpus and Ranking

With respect to our core contributions, we began in Chapter 4 by briefly describing a distributed crawling architecture for attaining a (generic) corpus of RDF data from the Web, exploiting Linked Data principles to discover new documents; we ran this crawler for 52.5 h over nine machines to retrieve 1.118 billion quadruples of RDF data from 3.985 million Web documents, constituting the evaluation corpus used for the later chapters. As such, we presented a scalable method for consumers to acquire a large corpus of RDF data from the Web, thus documenting the means by which our evaluation corpus was achieved, and thereafter presenting some high-level statistics to help characterise the corpus.

Thereafter, we applied a PageRank-inspired analysis of the sources in the corpus, deriving a set of ranking scores for documents which quantifies their (Eigenvector) centrality within the Web of Data (in 30.3 h using nine machines). These ranks are used in later chapters for (i) giving insights into the importance of various RDFS and OWL primitives based on a summation of the ranks of documents which use them in our corpus; and (ii) for computing and associating ranks with individual triples, which then serve as input into the annotated reasoning system. We noted that the core (RDF/RDFS/OWL) vocabularies and other popular (DC/FOAF/SKOS) vocabularies constituted the highest ranked documents.

Reasoning

In Chapter 5, we then described our method for performing reasoning—in particular, forward-chaining materialisation—with respect to a subset of OWL 2 RL/RDF rules which we deem suitable for scalable implementation.

We first discussed standard reasoning techniques and why they are unsuitable for our scenario, also motivating our choice of rule-based materialisation. We then introduced the newly standardised OWL 2 RL/RDF ruleset, discussing the computational expense and the potential of quadratic or cubic materialisation associated with certain rules, thus initially motivating the selection of a subset.

Continuing, we introduced the rationale and basis for distinguishing and processing terminological data separately during the reasoning process, formalising soundness and conditional completeness results and presenting a two-stage inferencing procedure which (i) derives a terminological closure and partially evaluates the program (ruleset) with respect to terminological data; (ii) applies the partially evaluated (assertional) program against the bulk of the corpus. We detailed and initially evaluated a number of novel optimisations for applying the assertional program, enabled by the partial evaluation with respect to terminological data.

Reuniting with our use-case, we introduced our notion of “A-linear” reasoning involving rules with only one assertional atom, giving a maximal size for materialised data possible therefrom; we identified the A-linear subset of OWL 2 RL/RDF as being suitable for our scenario, enabling linear complexity and materialisation *with respect to the assertional data* and—as also demonstrated by related approaches [Weaver and Hendler, 2009; Urbani et al., 2009]—enabling a straightforward distribution strategy whereby terminological knowledge is effectively made global across all machines, allowing these machines to perform assertional reasoning independently, and in parallel. Acknowledging the possibility of impudent terminological contributions by third parties on the Web, we introduced our approach for authoritative reasoning which conservatively considers only those terminological axioms offered by unambiguously trustworthy sources.

Finally, we presented evaluation of the above methods against our Linked Data corpus, providing an analysis of the terminological axioms used therein, validating our authoritative reasoning approach, analysing the proposed assertional program optimisations, presenting the size of the materialised data, and measuring the timing of the distributed tasks; in particular, using nine machines, we infer 1.58 billion raw triples (of which 962 million are novel and unique) in 3.35 h.

Annotated Reasoning

Recognising the possibility of noisy inferences being generated, in Chapter 6 we investigated an annotation framework for tracking metainformation about the input and inferred data during the reasoning process—metainformation which encodes some quantification of trust, provenance or data quality, and which is transformed and aggregated by the framework during reasoning. In particular, our primary use-case is to track ranking annotations for individual triples, which are subsequently used to repair detected inconsistencies in a parsimonious manner; additionally, we incorporate annotations for blacklisting malignant data or data sources, and metadata relating to authoritative reasoning.

As such, we first outlined our method for deriving ranks for individual triples in the input corpus: we loosely follow the approach of [Harth et al., 2009] whereby the rank of a triple is the summation of the ranks of documents in which it appears.

Continuing, we then formalised a generic annotated reasoning framework, presenting a number of reasoning tasks one might consider within such a framework, discussing various aspects of the framework and associated tasks with respect to scalability and growth of annotated materialisations—here, we eventually appealed to specific characteristics of our annotation domain which enable scalable implementation. We also introduced and discussed OWL 2 RL/RDF constraint rules which are used to detect inconsistency.

Moving towards our use-case, we used the ranks of document (computed above) to annotate triples with aggregated ranking scores—using nine machines, this process took 4.2 h. We then discussed extension of our distributed reasoning engine to incorporate annotations: using the same setup, annotated reasoning took 14.6 h including aggregation of the final results, producing 1.889 billion unique, optimal, annotated triples merged from input and inferred data. Concluding the chapter, we sketched a strategy for detecting and repairing inconsistencies (in particular, using the rank annotations) and discussed a distributed implementation thereof: using the same setup, detecting *and* repairing 301,556 inconsistencies—97.6% of which were ill-typed literals, with the remaining 2.4% given by memberships of disjoint classes—in the aggregated annotated corpus took 5.72 h.

Consolidation

Finally, in Chapter 7, we looked at identifying coreferent individuals in the corpus—individuals which signify the same real-world entity, but which are given different identifiers, often by different publishers. Given that thus far our reasoning procedures focussed on application of rules with only one assertional atom, and that rules supporting equality in OWL 2 RL/RDF contain multiple such atoms, we presented bespoke methods for handling the semantics of `owl:sameAs` in a scalale manner.

Firstly, we discussed the standard semantics of equality, and motivated our omission of `owl:sameAs` rules which affect terminology; we also motivated our canonicalisation approach, whereby one identifier is chosen from each coreferent set and used to represent the individual in the consolidated corpus—in particular, consolidation bypasses the quadratic materialisation mandated by the standard semantics of replacement, and can be viewed as a partial materialisation approach. We also presented statistics of our corpus related to naming, highlighting sparse reuse of identifiers across data sources.

We then began by detailing our distributed baseline approach whereby we only consider explicit `owl:sameAs` relationships in the data, which we load into memory and send to all machines; for our corpus, this approach found 2.16 million coreferent sets containing 5.75 million terms, and took 1.05 h on eight slave machines (including the consolidation step).

We extended this approach to include inference of additional `owl:sameAs` relationships using the reasoning approach of Chapter 5, as well as (inverse-)functional properties and certain cardinality constraints; this approach requires more on-disk processing and took 12.3 h on eight machines, identifying 2.82 million coreferent sets containing 14.86 million terms (unlike the baseline approach, a high percentage of these [60.8%] were blank-nodes).

Finally, acknowledging that some of the coreference we identify may be unintended despite the fact that our methods rely on the formal semantics of the data—i.e., that some subset of the identified coreference may be attributable to the inherent noise in the corpus or to unanticipated inferencing—we investigated using inconsistencies as indicators of defective consolidation, and sketched a bespoke method for repairing “unsatisfiable” sets of coreferent identifiers. Using eight slave machines, locating and repairing 523 unsatisfiable coreferent sets—and reflecting the reparations in the consolidated corpus—took 3.91 h.

Critique of Hypothesis

In light of what we have seen thus far, we take this opportunity to critically review the central hypothesis of this thesis as originally introduced in § 1.2:

> **Given a heterogeneous Linked Data corpus, the RDFS and OWL semantics of the vocabularies it contains can be (partially) leveraged in a domain-agnostic, scalable, Web-tolerant manner for the purposes of (i) automatically translating between (possibly remote) terminologies; and (ii) automatically resolving (possibly remote) coreferent assertional identifiers.**

Herein, Chapters 5 & 6 have translating assertional data between terminologies, and Chapter 7 has dealt with the resolution of coreference between assertional identifiers. We now discuss how the presented methods handle the three explicit requirements: (i) domain agnosticism, (ii) scalability, and (iii) Web-tolerance.

Domain Agnosticism

All of the methods presented in this thesis rely on a-priori knowledge from the RDF [Manola et al., 2004], RDFS [Hayes, 2004] and OWL (2) standards [Hitzler et al., 2009], as well as Linked Data principles [Berners-Lee, 2006]. As such, we show no special regard to any domain, vocabulary or data provider, with one exception: for consolidation, we require a manual blacklist of values for inverse-functional properties (see Table 7.5); although many of these could be considered domain-agnostic—for example, empty literals—values such as the SHA1 sum of `mailto:` are designed to counter-act malignant data within specific domains (in this case, values for the property `foaf:mbox_sha1sum`). Indeed, this blacklist constitutes additional a-priori knowledge outside of the remit of the hypothesis, although we view this as a minor transgression. However, our blacklist is a reminder that as Linked Data diversifies—and as the motivation and consequences of active spamming perhaps become more apparent—mature consumers may necessarily have to resort to heuristic and domain-specific counter-measures to ensure the effectiveness of their algorithms, analogously to how Google (apparently) counteracts deliberate spamming on the current Web.

Similarly, Linked Data consumers may find it useful to enhance the generic "core" of their system with domain-specific support for common or pertinent vocabularies. In our own primary use-case—the Semantic Web Search Engine (SWSE)[1]—we have manually added some popular properties (from which `rdfs:label` values cannot be inferred) to denote labels for entities, including, for example, `dc:title`.[2] Likewise, the Sig.ma search interface [Tummarello et al., 2009] avoids displaying the values of selected properties—e.g., `foaf:mbox_sha1sum`—which it deems to be unsightly to users. Aside from user-interfaces, for example, Shi et al. [2008] and Sleeman and Finin [2010] have looked at FOAF-specific heuristics for consolidating of data, Kiryakov et al. [2009] manually select what they deem to be an interesting subset of Linked Data, etc.

Clearly, domain agnosticism may not be a strict requirement for many Linked Data consumers, and especially in the current "bootstrapping" phase, more convincing results can be achieved with domain-specific tweaks. However, for popular, large-scale consumers of heterogeneous Linked Data, neutrality *may* become an important issue: showing special favour to certain publishers or vocabularies may be looked upon unfavourably by the community. On a less philosophical level, improving results by domain agnostic means is more flexible to changes in publishing trends and vocabularies. In any case, as more appealing applications emerge for Linked Data, publishers will naturally begin tailoring their data to suit such applications; similarly, one can imagine specific, high-level domain vocabularies—such as the Fresnel vocabulary [Pietriga et al., 2006] which allows for publishing declarative instructions on how RDF should be rendered—emerging to meet the needs of these applications.

[1] An online prototype is available at `http://swse.deri.org/`.

[2] Much like the blacklist, we do this with some reluctance—within SWSE, we wish to strictly adhere to domain-independent processing of data.

Scalability

To ensure reasonable scale, we implement selected subsets of standard reasoning profiles, and use non-standard optimisations and techniques—such as separating terminological data from assertional data, and canonicalising equivalent identifiers—to make our methods feasible at scale. Our implementations rely primarily on lightweight in-memory data structures and on-disk batch processing techniques involving merge-sorts, scans, and merge-joins. Also, all of our methods are designed to run on a cluster of commodity hardware, enabling some horizontal scale: adding more machines typically allows for more data to be processed in shorter time. We have demonstrated all of our methods to be feasible over a corpus of 1.118 billion quadruples recently crawled from Linked Data.

With respect to reasoning in general, our scalability is predicated on the segment of terminological data being relatively small and efficient to process and access; note that for our corpus, we found that ~0.1% of our corpus was what we considered to be terminological. Since all machines currently must have access to all of the terminology—in one form or another, be it the raw triples or partially evaluated rules—increasing the number of machines in our setup does not increase the amount of terminology the system can handle efficiently. Similarly, the terminology is very frequently accessed, and thus the system must be able to service lookups against it in a very efficient manner; currently, we store the terminology/partially-evaluated rules in memory, and with this approach, the scalability of our system is a function of how much terminology can be fit on the machine with the smallest main-memory in the cluster. However, in situations where there is insufficient main memory to compute the task in this manner, we believe that given the apparent power-law distribution for class and property memberships (see Figures 4.6(b) & 4.6(c)), a cached on-disk index would work sufficiently well, enjoying a high-cache hit rate and thus a low average lookup time.

Also, although we know that the size of the materialised data is linear with respect to the assertional data, another limiting factor for scalability is how much materialisation the terminology mandates—or, put another way, how deep the taxonomic hierarchies are under popularly instantiated classes and properties. For the moment, with some careful pruning, the volume of materialised data roughly mirrors the volume of input data; however, if, for example, the FOAF vocabulary today added ten subclasses of `foaf:Person`, the volume of authoritatively materialised data would dramatically increase.

Also related to the terminology, we currently use a master machine to coordinate global knowledge which may become a bottleneck in the distributed execution of the task, depending on the nature and volume of the data involved; one notable example of this was for the distributed ranking, where the PageRank analysis of the source-level graph on the master machine proved to be a significant bottleneck. Admittedly—and appealing to the current exploratory scope—our methods currently do not make full use of the cluster, where many of the operations currently done by the master machine could be further parallelised (such as the PageRank iterations; e.g., see [Gleich et al., 2004]). We consider this as potential future work.

Some of our algorithms require hashing on specific triple elements to align the data required for joins on specific machines; depending on the distribution of the input identifiers, hash-based partitioning of data across machines may lead to load balancing issues. In order to avoid such issues, we do not hash on the predicate position of triples or on the object of `rdf:type` triples given the distribution of their usage (see Figures 4.6(b) & 4.6(c)—particularly the x-axes): otherwise, for example, the slave machine that receives triples with the predicate `rdf:type` or object `foaf:Person` would likely have significantly more data to process than its peers (see Table 4.4). Although elements in other positions of triples also demonstrate a power-law like distribution (see Figure 4.6(a)), the problem of load-balancing is not so pronounced—even still, this may become an issue if, for example, the number of machines is significantly increased.

Relatedly, many of our methods also rely on external merge-sorts, which have a linearithmic complexity $O(n * \log(n))$; moving towards Web-scale, the $\log(n)$ factor can become conspicuous with respect to performance. From a practical perspective, performance can also depreciate as the number of on-disk sorted batches required for external merge-sorts increases, which in turn increases the movement of the mechanical disk arm from batch to batch—at some point, a multi-pass merge-sort may become more effective, although we have yet to investigate low-level optimisations of this type. Similarly, many operations on a micro-level—for example, operations on individual entities or batches of triples satisfying a join—are of higher complexity; typically, these batches are processed in memory, which may not be possible given a different morphology of data to that of our current corpus.

Finally, we note that we have not addressed dynamicity of data: our methods are primarily based on batch processing techniques and currently assume that the corpus under analysis remains static. In our primary use-case SWSE (§ 2.4), we envisage a cyclic-indexing paradigm whereby a fresh index is being crawled, processed and indexed on one cluster of machines whilst a separate cluster offers live queries over the most recent complete index. Still, our assumption of static data may be contrary to the requirements of many practical consumer applications wishing to consume dynamic sources of information, for which our current performance and scalability results may not directly translate. However, we still believe that our work is relevant for such a scenario, subject to further research; for example, assuming that the majority of data remain static, a consumer application could use our batch processing algorithms for this portion of the corpus, and handle dynamic information using smaller-scale data structures. Similarly, for example, assuming that the terminological data is sufficiently static, our classical reasoning engine can easily support the addition of new assertional information. Still, the applicability of our work for dynamic environments is very much an open (and interesting) research question.

Web Tolerance

With respect to Web-tolerance, we (i) only consider authoritative terminology, (ii) consider the source-level graph when performing ranking, (iii) blacklist common vacuous values for inverse-functional properties, (iv) avoid letting consolidation affect terminology, predicates and values of `rdf:type`, and (v) use PageRank and concurrence similarity-measures to debug and repair detected inconsistencies. We have demonstrated these techniques to make non-trivial forms of materialisation and consolidation feasible over our corpus (collected from 3.985 million sources).

However, we are not tolerant to all forms of publishing errors and spamming. In particular, we are still ill-equipped to handle noise on an assertional-level, where we mainly rely on inconsistency to pinpoint such problems, and where many types of noise may not be symptomised by inconsistency. In fact, we found only modest amounts of inconsistency in the corpus, mainly due to invalid datatypes and some memberships of disjoint classes—many noisy (but consistent) inferences and coreference relations can persist through to the final output.

Similarly, we do not directly tackle the possibility of deliberate spamming on an assertional level—needless to say that considering all information provided about a given entity from all sources is vulnerable to the spamming of popular entities with impertinent contributions. However, we do track the source of data, which can subsequently be passed on to the consumer application.[3] Thereafter, a consumer application can consider using bespoke techniques—perhaps based on something similar to our notion of authority, or the presented links-based ranking—to make decisions on the value and trustworthiness of individual contributions.

In summary, it is very difficult to pre-empt all possible forms of noise and spamming, and engines such as Google have adopted a more reactive approach to Web-tolerance, constantly refining their algorithms to better cope with the inherent challenges of processing Web data. Along similar lines, we have demonstrated reasoning and consolidation methods which can cope with many forms of noise present on today's Web of

[3] For rules with only one assertional patten (as per our selected subset of OWL 2 RL/RDF) we can optionally assign each assertional inference the context of the assertional fact from which it is entailed.

Data, perhaps serving as a foundation upon which others can build in the future (as necessary).

Future Directions

In this thesis, we have demonstrated that non-trivial reasoning and consolidation techniques are feasible over large-scale corpora of current Linked Data in the order of a billion triples. We now look at what we feel to be important future directions arising from the work presented in this thesis. In particular, we identify a number of high-level areas for future works which we believe to be:

1. of *high-impact*, particularly with respect to Linked Data publishing;
2. *relevant* for integrating heterogeneous Linked Data corpora;
3. *feasible* for large-scale, highly-heterogeneous corpora collected from unvetted sources;
4. *challenging and novel*, and thus suitable for further study in a research setting.

The five areas we identify are as follows:

1. identifying a **"sweet-spot" of reasoning expressivity**, taking into account computational feasibility as well as adoption in Linked Data publishing;
2. exploring the possibility of **publishing rules within Linked Data**, where rules offer a more succinct and intuitive paradigm for axiomatising many forms of entailment when compared with RDFS/OWL;
3. investigating more robust/conservative criteria for **trustworthiness of assertional data**, where we see a need for further algorithms which tackle impudent third-party instance data, or, e.g., erroneous `owl:sameAs` mappings;
4. researching **statistical or machine learning approaches** for performing reasoning and consolidation over Linked Data, which leverage the ever increasing wealth of RDF Web data becoming available;
5. designing, creating and deploying better **evaluation frameworks for realistic and heterogeneous Linked Data**.

Reasoning Expressivity: Finding the "Sweet-spot"

With respect to reasoning expressivity, there are thus two primary dimensions to consider: (i) how computationally feasible is that expressivity of reasoning; (ii) what expressivity is commonly used by (and/or is useful for) Linked Data vocabularies.

With respect to computational feasibility, in this thesis we apply materialisation with respect to a scalable subset of OWL 2 RL/RDF rules which enables an efficient and distributable inference strategy based on a separation of terminological knowledge. In particular, we restrict our rules to those which have zero or one assertional atoms in the body of the rule, which (i) ensures that the amount of materialised data stays linear with respect to the assertional data in the corpus, and (ii) allows for a distribution strategy requiring little co-ordination between machines. With respect to extending our approach to support a fuller subset of OWL 2 RL/RDF, we note that certain rules which have multiple assertional atoms in the body do not affect our current guarantees on how much data are materialised: these are rules where each atom in the head contains at most one variable not appearing in any T-atom, where an example is cls-svf1:

$$(?u,\ a,\ ?x) \leftarrow \underline{(\texttt{?x, owl:someValuesFrom, ?y}),(\texttt{?x, owl:onProperty, ?p})},\ (?u,\ ?p,\ ?v),(?v,\ a,\ ?y)\ .$$

Since the head variable ?x also appears in the T-atoms of the rule, ?u is the only head variable not appearing in a T-atom, and so the number of inferences given by this rule is bounded by the number of groundings for the pattern (?u, ?p, ?v), and so is bounded by the amount of assertional data. However, such rules would still require amendment to how we distribute our reasoning, perhaps following the works of Urbani et al. [2010] or Oren et al. [2009b], etc. However, rules for other OWL primitives—such as owl:TransitiveProperty—introduce the immutable possibility of quadratic (or even cubic) growth in materialisation. Thus, a pure materialisation approach to such inferencing is perhaps not appropriate, where, instead, partial materialisation may prove more feasible in the general case: for example, one could consider an approach which materialises the partial closure of transitive chains in the data, and applies backward-chaining at runtime to complete the chains.[4] Generalising the problem, an interesting direction for future research is to investigate coherent cost models with respect to supporting inferences by means of forward-chaining and backward-chaining, thus allowing to optimise the inferencing strategy of a given system in its native environment.

Aside from computational feasibility, the "sweet-spot" in expressivity is also predicated on the use of RDFS and OWL within popular Linked Data vocabularies. For example, we currently support primitives—such as owl:hasValue, owl:intersectionOf, etc.—which we found to have scarce adoption amongst the vocabularies in our corpus; similarly, for consolidation we support owl:cardinality and owl:maxCardinality axioms which allow for inference of owl:sameAs relations and which add an additional computational expense to our methods, but which gave no results for our corpus. In general, our survey of Linked Data vocabularies showed an inclination towards using those RDFS and OWL primitives whose axioms are expressible in a single triple, and—with the possible exception of owl:unionOf—a disinclination to use OWL primitives whose axioms require multiple triples, such as complex class descriptions using RDF lists, or those involving OWL restrictions, etc. We believe that there is now sufficient adoption of RDFS and OWL in the Wild to be able to derive some important insights into what parts of the RDFS and OWL standards are being used, and to what effect. Having provided some initial results in this thesis, we would welcome further investigation of RDFS and OWL adoption, with the possible goal of identifying a subset of (lightweight) primitives recommended for use in Linked Data, along with associated best practices and rationale backed by the empirical analyses.

Thus, the sweet-spot in expressivity should consider computational feasibility (including ease of implementation to support the required inferencing) and the needs of publishers. In this thesis, we have contributed some empirical evidence which already suggests that publishers favour the use of those lightweight primitives which are supported by our scalable subset of OWL 2 RL/RDF.

Terminology vs. Rules

In our scenario, consistency cannot be expected: thus, we claim that tableau-based approaches are not naturally well-suited to our requirements. Along these lines, our framework is based on monotonic rules, where we compile Linked Data vocabularies into T-ground rules, such as:

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

Accordingly, our framework is also compatible with generic RDF rules such that can be expressed in a number of declarative Semantic Web rule languages, including N3 [Berners-Lee, 1998a], SWRL [Horrocks et al., 2004] or RIF Core [Boley et al., 2010].[5] Interestingly, such rule languages cover a number of "blind-spots" in OWL expressivity; for example, the inferencing encoded by the rule:

[4] Note that, in effect, we use such a "partial-materialisation" approach for our consolidation.

[5] By generic RDF rules, we mean Horn clauses which only use RDF atoms and whose variables are range-restricted. Note that the stated rule languages can express more complex forms of rules which we do not explicitly discuss here.

$$(\texttt{?x}, \texttt{ex:hasYoungerSister}, \texttt{?y}) \leftarrow (\texttt{?x}, \texttt{ex:olderThan}, \texttt{?y}),(\texttt{?x}, \texttt{ex:hasSister}, \texttt{?y})$$

is not expressible in OWL 2 DL.[6] Many other forms of inference are much more easily encoded as rules, rather than RDFS/OWL; for example, consider the rule:

$$(\texttt{?x}, \texttt{ex:hasBrother}, \texttt{?y}) \leftarrow (\texttt{?x}, \texttt{ex:hasSibling}, \texttt{?y}),(\texttt{?y}, \texttt{a}, \texttt{ex:Male})\ .$$

whose inferences can only be modelled in OWL 2 using the following prolix terminology:

$$\begin{gathered}
(\texttt{ex:Male}, \texttt{rdfs:subClassOf}, \texttt{\_:hasSelfMale}),\\
(\texttt{\_:hasSelfMale}, \texttt{owl:hasValue}, \texttt{"true"\^{}\^{}xsd:boolean}),\\
(\texttt{\_:hasSelfMale}, \texttt{owl:onProperty}, \texttt{ex:selfMale}),\\
(\texttt{ex:hasBrother}, \texttt{owl:propertyChainAxiom}, \texttt{\_:listOne}),\\
(\texttt{\_:listOne}, \texttt{rdf:first}, \texttt{ex:hasSibling}), (\texttt{\_:listOne}, \texttt{rdf:rest}, \texttt{\_:listTwo}),\\
(\texttt{\_:listTwo}, \texttt{rdf:first}, \texttt{ex:selfMale}), (\texttt{\_:listTwo}, \texttt{rdf:rest}, \texttt{rdf:nil})\ .
\end{gathered}$$

or, in more legible Turtle syntax:

```
ex:Male rdfs:subClassOf [ owl:hasSelf true, owl:onProperty ex:selfMale ] .
   ex:hasBrother owl:propertyChainAxiom ( ex:hasSibling ex:selfMale ) .
```

We note that the required terminology is unintuitive with respect to its intention, and must encode some "auxiliary" definitions to achieve the desired inferences. In general, we believe that the simple `IF-THEN` structure of rules is a more direct and intuitive formalism than the RDFS and OWL languages, and would thus be more amenable to adoption by a wider community of practitioners.

This begs the question: would a pure rule-based paradigm better suit the Linked Data community than the current RDFS and OWL paradigm of publishing the semantics of terms in vocabularies? On the other side of the argument, we note that RDFS and OWL are *descriptive* as well as *prescriptive*: as well as prescribing the entailments possible through a given set of terms, the RDFS and OWL languages also allow for giving a direct and rich RDF description of those terms and their inter-relations. Similarly, `owl:sameAs` (and `owl:differentFrom`) offer a terse relation for asserting (or rejecting) coreference such that can be directly embedded into the given RDF data. In addition, the semantics of more expressive constructs such as `owl:disjointUnionOf` or classes with high-cardinality restrictions may prescribe entailments which require complex rulesets to axiomatise, although we note that the use of such primitives is uncommon in current Linked Data.

Still however, it seems that rules and vocabularies offer complementary approaches—as was the motivation behind various proposals such as SWRL, DLP [Grosof et al., 2004], Datalog± [Calì et al., 2010] and OWL 2 RL [Grau et al., 2009]—although the focus thus far for Linked Data has largely been on vocabularies. Notably, rule-based approaches such as SHOE [Heflin et al., 1999], N3 and SWRL pre-date the Linked Data principles, but, to the best of our knowledge, have yet to see significant adoption on the Web; similarly, various proposals for encoding rules as SPARQL CONSTRUCT queries [Polleres, 2007; Schenk and Staab, 2008; Bizer and Schultz, 2010] or for encoding constraints as SPARQL ASK queries[7], have yet to see adoption in the Wild. In particular, whilst there are various best-practices regarding how to publish vocabularies on the Web [Miles et al., 2006], there are few guidelines available regarding how to publish rules on the Web. Although there is ongoing work in the W3C on an initial proposal for publishing RIF rules as RDF [Hawke, 2010], and community proposals for describing SPARQL/SPIN rules as RDF[8], the resulting

[6] This inference involves role conjunction (a.k.a. property/role intersection) whose inclusion into OWL 2 DL would lead to a higher complexity class (see, e.g., [Glimm and Kazakov, 2008]).

[7] `http://www.spinrdf.org/`; retr. 2011/02/16

[8] For example, see `http://www.spinrdf.org/spin.html#spin-constraint-ask`; retr. 2011/02/16

documents (necessarily) require use of complex nested RDF structures and thus are rather unintuitive and difficult to read.

In summary, we see that there is clear motivation to publish rules on the Web—possibly alongside traditional vocabularies—where there are already some existing proposals in the area, but as of yet no notable adoption. Along these lines, an interesting research direction would be to look at use of rules in the Wild, and to investigate how declarative rules can be encoded, published and shared across the Web in a manner propitious to the Linked Data community. Various non-trivial questions then arise with respect to how vocabularies and published rules interplay, what expressivity rules should allow, what form of negation should be supported [Wagner, 2003; Polleres et al., 2006], whether or not rules should follow the Open World Assumption and the lack of a Unique Name Assumption as per OWL [de Bruijn et al., 2005a, 2006; Motik et al., 2009a],[9] how rules can be trusted or interlinked, etc. Although there is already a significant body of literature (partially) tackling these questions, we feel that further research within a realistic Linked Data setting would have considerable practical impact.

Assertional Authority

Currently, although we critically examine the source of terminological data using our notion of authority, we assume a certain level of good-faith with respect to the provision of assertional data in our corpus. For example, we would consider any "consistency-preserving" `owl:sameAs` relation as grounds for consolidating two resources, no matter where the relation is given. Even where such mappings are given "in earnest"—and as we have seen ourselves in this thesis, and as has been discussed in detail by Halpin et al. [2010a]—overly liberal (mis)use of `owl:sameAs` can often occur, leading to noise. Further still, given sufficient commercial adoption of Linked Data, motivation for spamming becomes more substantive, and our assumption of good faith quickly becomes naïve.

Hence, we believe that different forms or interpretations of `owl:sameAs` relations are needed; in particular, non-symmetric `owl:sameAs` relations (or interpretation of `owl:sameAs` relations as being non-symmetric) would offer a directional "import" mechanism whereby one resource description imports the data for an equivalent resource provided by another source. This would result in entities having different "views" depending on which source is trusted, and which descriptions are recursively imported: again, such a version of `owl:sameAs` would naturally reduce the ability for ad-hoc alignment of resources across the Web, but would offer better tolerance to impudent third-party contributions. Along these lines, McCusker and McGuinness [2010] have discussed eight granular variants of `owl:sameAs` which are alternatively reflexive/non-reflexive, symmetric/non-symmetric, transitive/non-transitive.

However, the challenge of what assertional data should be trusted extends beyond `owl:sameAs`, where third-parties can still simply re-use external URIs to provide spurious claims about arbitrary resources. Hence, tracking the source of information is of vital importance, allowing users to verify the results presented to them. However, oftentimes, tracking the combination of sources of intermediary data involved in generating a given result may be computationally infeasible, and may be difficult for a user to review or verify. Thus, automated processes for immediately identifying and rejecting untrustworthy data are of vital importance, in particular for lightening or removing the burden of result-verification for users. Our annotated reasoning approach offers some initial results in this direction, and we would again welcome more focussed research into this area—not necessarily restricted to inferencing, but also looking at other specific areas such as querying, or perhaps looking at the general problem of computing, combining and tracking notions of provenance and trust, perhaps building upon the related works discussed in § 6.5.

[9] These issues are particularly relevant when discussing constraints in a (semi-)closed setting, where often publishers want to check for missing data (a task complicated by the OWA), or for setting a maximum number of values that can be assigned to a given property (a task complicated by the lack of UNA).

Statistical Reasoning/Consolidation Methods

Herein, we have primarily focussed on symbolic approaches to perform reasoning and consolidation. However, given that we deal with vast amounts of often loosely co-ordinated Web data, such algorithms can also be informed by patterns observable in the data, rather than considering explicit formal assertions alone. Along these lines, we currently employ links-analysis techniques for determining a level of "trustworthiness" for triples (§ 6.2.3), and have introduced a "concurrence measure" which quantifies a form of pair-wise similarity between resources based on a statistical model (§ D). However, we envisage other areas in which statistical approaches can complement our symbolic approach for reasoning and consolidation, in particular (but not solely), to complement inconsistency detection for identifying noisy inferences, or for disambiguating resources initially found to be coreferent according to the formal semantics of the data. More ambitiously, *inductive* reasoning or consolidation approaches—which attempt to learn and apply new rules from observed macro-patterns in the data—may prove fruitful: for example, machine learning approaches for reasoning or consolidation could use the results given by symbolic approaches as training sets for building generic models useful for finding novel results and improving recall.

Such approaches largely echo the viewpoint of Halevy et al. [2009]:

> *"Problems that involve interacting with humans* [...] *have not proven to be solvable by concise, neat formulas like $F = ma$. Instead, the best approach appears to be to embrace the complexity of the domain and address it by harnessing the power of data: if other humans engage in the tasks and generate large amounts of unlabeled, noisy data, new algorithms can be used to build high-quality models from the data."*
>
> —**[Halevy et al., 2009]**

In summary, given enough raw data, seemingly complex problems can be solved by building models which fit the data, and thereafter applying these models to specific instances of the problem. However, with respect to the current Web of Data, we identify two major high-level challenges faced by any such approach.

Firstly, although the current Web of Data contains (at least) tens of billions of statements, these are provided by relatively few domains, where in our crawl we found 783 domains hosting RDF/XML—currently, the bulk of RDF documents comprising the Web of Data come from a relatively small number of high-volume exporters. Although more and more data-providers are emerging, in our experience, statistical models must reflect the current skew in RDF provided by different exporters: otherwise, models will be heavily influenced by the nature of the data given by those few very large exporters and will not accurately reflect the nature of the data given by many small exporters. For example, consider an approach for disambiguating resources which looks at how functional certain properties are, where from the data we observe that people have on average 1.0001 values for `foaf:weblog`—now, if we find a consolidated entity with two values for `foaf:weblog`, we may consider this anomalous and decide that the consolidation was incorrect. However, the value 1.0001 is heavily influenced by large, highly-uniform FOAF exporters which only allow one weblog to be assigned to each of its users; conversely, the few most influential and richly described entities in the data may have multiple weblogs (e.g., work, personal, new location, etc.), where the model is no longer appropriate, and where disambiguation should not be applied. Therefore, we believe that a statistical approach should either (i) build and apply individual models for different *contexts* (e.g., different PLDs or groups of exporters); or (ii) build and apply one global model whereby each context has a limited influence. Thus, aside from looking at specific statistical techniques for reasoning and consolidation, high-level research on contextually scoping and combining generic statistical models for Linked Data could potentially be very interesting.

Secondly, there are inherent difficulties with respect to how the precision and/or recall of such an approach can be evaluated. This ties in with the next area of research with respect to how systems operating over realistic Linked Data can be evaluated in a meaningful manner, particularly with respect to the quality of results achieved.

Evaluation Framework for Linked Data

The evaluation of the *quality* of our results is still an open question, especially given that the current proliferation of Linked Data is a relatively new phenomenon, where gold standards and objective benchmarks have yet to be defined.[10] Most Semantic Web research currently relies on synthetic evaluation frameworks, including the Lehigh University Benchmark [Guo et al., 2005], the University Ontology Benchmark [Ma et al., 2006], SP^{2}Bench [Schmidt et al., 2009b], the Berlin SPARQL Benchmark [Bizer and Schultz, 2009], etc., which define a correct, clean, homogeneous, artificial and simplistic terminology and associated assertional data according to generative algorithms which result in uniform and predictable morphology; thereafter, a small set of target queries are manually selected based on generic criteria. Thus, we believe that such benchmarks are not representative of the complexity and noise inherent in real-world Linked Data, and results achieved using such synthetic frameworks—in terms of scalability, performance, precision, recall, quality of results, etc.—do not necessarily translate into those that would be achieved for scenarios such as ours. Conversely, when using real-world Linked Data for evaluation, in the absence of a gold standard we often rely on subjective experience, manual inspection of small samples, analysing prominent examples, looking at secondary measures and related statistics, etc., to gain insights into the effectiveness of our algorithms.

The methods we present in this thesis are heavily rooted in the Semantic Web standards and on the Linked Data principles, and as such are predicated on the correct use of said by the publishers contributing to the corpus under analysis. However, the assumption of correctness is admittedly naïve, where we have shown that the inherent noise present in such data sometimes leads to incorrect or unintended consequences in our reasoning and consolidation approaches. As such, we have incorporated analyses which critically examine the source of data during reasoning. We have also presented methods to detect and repair inconsistencies caused by our methods, but have also shown that Linked Data vocabularies currently only provide sparse axiomatisations of what they consider to be inconsistent: besides trivially invalid datatype literals, we find relatively few detections of inconsistency (further, we note that FOAF have recently removed disjointness axioms between `foaf:Person` and `foaf:Document` which gave many of the inconsistency detections encountered in our evaluation).

In general, we currently cannot pinpoint the *precision* of our methods, whereby we cannot directly ascertain which inferences could be considered noise, or which consolidated resources are incorrect. Similarly, we cannot determine an accurate figure for *recall*, whereby we cannot know the percentage of inferences we find, or the percentage of coreferent resources which we cannot detect. Hitzler and van Harmelen [2010] have also argued for applying measures of precision and recall for reasoning systems, noting that soundness and completeness with respect to the RDF(S)/OWL standards is often infeasible, where, instead, approximate and/or incomplete reasoning approaches may offer appealing performance benefits: however, they view sound and complete reasoning with respect to RDF(S)/OWL as a "gold standard", thus again implicitly assuming that the data are correctly published according to those standards, and that reasoning will give all possible desirable results. However, we believe that purely quantitative precision and recall measures are not sufficient, particularly for materialisation where many consequences could be considered purely inflationary and expensive for a consumer application to support. Thus, we believe that *qualitative* measures are also of vital importance—with respect to the importance of the inferences a system includes/excludes or the

[10] A promising initiative in this direction is the new SEALS EU infrastructures project `http://www.seals-project.eu/` which deals with benchmarking and evaluating Semantic Web tools—the project is still in its ramp-up phase, and we hope to see benchmarks specifically targeting Linked Data use-cases.

importance of the coreference a system finds/misses—where a system which returns fewer but higher-quality results may often be preferred.

Looking towards the (near) future, we thus see a great need for more realistic Linked Data evaluation frameworks to test the quality of results produced by systems such as ours. Such frameworks will enable higher-quality and more relevant research for real-world Linked Data, perhaps leading to new results or insights on current wisdom and standard approaches, or perhaps also allowing for more convincing research into speculative and non-standard approaches, such as the statistical methods previously discussed.

The creation and maintenance of such a benchmark is indeed a rich research challenge in and of itself; it is difficult to see how a synthetic benchmark can appropriately reflect the complexities of real-world Web data, whereby it may be more shrewd to simply use an agreed upon corpus of real-world data. In fact, for the past three years we have been crawling the data for ISWC's Billion Triple Challenge: a show-case for scalable tools that must operate over the given corpus. Although the intent of these datasets has not been academic, they have been used in a number of scientific papers [Erling and Mikhailov, 2009; Hogan et al., 2009b; Urbani et al., 2009; Schenk et al., 2009] to demonstrate scale or applicability of methods for realistic data. Thus, the demand for a normative and agreed-upon benchmark of realistic Linked Data seemingly already exists.

A major challenge thereafter is defining the various gold standards needed for different algorithms to evaluate their results; with respect to reasoning and consolidation, we acknowledge that in many cases, whether certain inferences or coreferences are considered correct or incorrect, or of high or low quality, is often inherently subjective [Halpin et al., 2010a], difficult to quantify, and can vary according to the end consumer application. As such, the notion of a "universal gold standard" may not even exist, let alone be computable; creating a gold-standard for a small sample of realistic data may be sufficient for certain types of algorithms, but may not be suitable for techniques relying on statistical analyses or some form of inductive learning—that is, techniques which require large amounts of data upon which to build their models.

One straight-forward solution for estimating the precision of a system is to randomly sample results, and have the sample be manually verified by human experts. More ambitiously, a system could investigate using the "wisdom of crowds" to verify results, the challenge here being to attract a critical mass of users to interact with the system (a general challenge facing Linked Data systems), perhaps using games or other impetuses known from the social and behavioural sciences.

Measuring recall and quality in such settings is also an intrinsically difficult task. One possible solution is to use a common benchmark dataset and create a framework for comparing the results of systems with similar goals; thus, although no gold-standard is made available, different approaches can be effectively compared by analysing the observed deltas in computed results, manually sampling and verifying a small subset of the relevant deltas.

Although an inherently challenging area, by removing the reliance on synthetic benchmarks, we believe that a suitable framework for benchmarking over realistic data will foster a higher volume of more ambitious, high-quality research, relevant to the current requirements of Linked Data practitioners, and as such, is of critical importance moving forward.

Concluding Remarks

For my final remarks, I present some personal opinions and subjective observations.

At the time of writing, the Semantic Web community has been around in one form or another for just over a decade. Born in a primordial soup of vision statements—by the likes of Berners-Lee [1998b], van Harmelen and Fensel [1999] Decker et al. [2000], Motta [2000], Berners-Lee et al. [2001], Finin and Joshi [2002]–and research works—by the likes of Heflin et al. [1999], Hendler and McGuinness [2000], Staab et al. [2000], Fensel et al. [2001], Sintek and Decker [2002]—the Semantic Web has long taken a "top-down" approach, epitomised by the prevalence of the ethereal Semantic Web layercake: a macro-level vision of a Web that can gracefully overcome the current information overload experienced by its users, represented as a stack of components.[11] In antithesis to this approach, *Gall's Law* states that:

> *"A complex system that works is invariably found to have evolved from a simple system that worked. The inverse proposition also appears to be true: A complex system designed from scratch never works and cannot be made to work. You have to start over, beginning with a working simple system."*
>
> **—Gall's Law**

Perhaps in line with this (informal) hypothesis, appearances of the layercake are becoming more and more rare—perhaps it has merely served its purpose.

In any case, the Linked Data community has had success taking a different tack by actively promoting a simpler, more "bottom-up" approach focussing on lower levels of the layercake's stack. In terms of evangelism—and again antithetical to the layercake—the Linked Data principles and Linked Open Data star scheme emphasise the micro over the macro, with the principles giving individual publishers clear guidelines to follow, and the star scheme giving a tangible bottom-up rationale for publishing Linked Data. Thereafter, proponents hope that adoption of the micro principles will self-organise into a usable and useful macro Web of Data in a form of *emergence*.

Indeed, as we have seen, this Web of Data has already been partially realised, and represents an excellent foundation upon which to build. Moving forward, it seems that the most important ingredient currently missing is *consumers*: we need applications to actively demonstrate the unique potential of this new publishing paradigm, and in particular, the benefit of having *interlinked* and *decentralised* data, demonstrating that the Web of Data is greater than the sum of its parts. Without such consumers, there is the danger that Linked Data will lose its current momentum and stagnate; without the latter two elements—and although RDF may become a popular standardised data-model for exchange—there will be no incentive for publishers to link to external sources, or perhaps even to use agreed-upon vocabularies, and the Web of Data may devolve into an "Archipelago of Datasets". Conversely, the advent of such applications will not only refine the requirements for future Linked Data publishing and research, but will also incentivise the publication of higher-quality and more voluminous Linked Data, and perhaps ultimately engage mainstream adoption. To a certain extent, publishing and consuming can only float upwards together: they are inherently tethered to each other. Although early adopters and consumers are emerging—in the form of various online communities, corporations, governmental agencies, etc.—there is still much work to be done.

In the short term, the Semantic Web research community should not only build upon its successes in getting further Linked Data published and current Linked Data improved, but also focus on lowering the barriers of entry for applications to consume the resulting Web of Data. Along these lines, research initiatives—such as the recent Consuming Linked Data Workshop [Hartig et al., 2010] and the more established Semantic Web Challenges and Billion Triple Challenges at ISWC[12]— and less research-oriented events—such as various

[11] One can quickly get an idea of the layer-cake variants by typing "'`semantic web layer cake`" into any popular Web search engine for images: perusing the various visions for the future of the Web, one may encounter a 3-dimensional "cake" complete with a peripheral Linked Data tower (`http://bnode.org/media/2009/07/08/semantic_web_technology_stack_small.png`), an actual "cake" replete with icing (`http://leirdal.net/blog/wp-content/uploads/2010/06/semanticcake2.jpg`), and a rather more dystopian view of the matter (`http://www.flickr.com/photos/danbri/428172848/lightbox/`); all retr. 2011/01/22. One may also note the lack of components for newer technologies such as RDFa or GRDDL [Hendler, 2009].

[12] See `http://challenge.semanticweb.org/`; retr. 2010/12/28

(Linked) Open Data Hackathons[13]—are very much pertinent, and will hopefully give rise to new techniques and software to aid or inspire consumers.

Interestingly, the bottom-up Linked Data approach and top-down traditional Semantic Web approach are due to meet somewhere in the layercake: Linked Data will—in its own time—find its own motivation for incorporating techniques from the higher strata of the stack. As motivated in this thesis, there is already a palpable need for RDFS/OWL reasoning when querying (or otherwise consuming) heterogeneous Linked Data corpora; thus, research into scalable reasoning techniques—applicable over real-world Linked Data—may help remove some of the current barriers to adoption and enable new applications. I can only hope that the work presented in this thesis will represent a valuable contribution to this area—it is important to note that the motivation, requirements, and directions of the research presented herein have been practically informed by our SWSE use-case for searching and browsing Linked Data.

As the bottom-up and top-down efforts begin to mingle, discussions with respect to refining the Semantic Web standards may again be informed by future developments; here, *Sowa's Law of Standards* feels (informally) appropriate:

> *"Whenever a major organization develops a new system as an official standard for* X*, the primary result is the widespread adoption of some simpler system as a de facto standard for* X*."*

—Sowa's Law of Standards

If we are to engage the broader Web community, we will need to provide more laconic standards, which are more intuitive to developers, and more tangible to publishers—perhaps the greatest enemy currently facing adoption is our tendency to try and do too much all at once. Along these lines, discussions on possibly revising or simplifying RDF [Wood et al., 2010] are of huge important in light of recent developments. Further down the road, we may be in a better position to cast a critical eye at the RDFS and OWL standards—more succinct standards may emerge, informed by previous theoretical works and perhaps exploratory works such as this, but with emphasis on *terseness* as appropriate for Linked Data adopters.

In conclusion, thanks to the bottom-up efforts of the Linked Data community, the future of the Semantic Web looks bright. Although we have yet to achieve mainstream adoption, it certainly seems a lot more realisable than two or three years ago. In order to get there, we may have to wander some ways from the envisaged Semantic Web road-map and abandon to ruin some parts of the towering layercake, we may have to deprecate parts of the familiar standards and compromise our academic ideologies in favour of pragmatic results, we may have to adapt and we may have to get our hands dirty. In any case, we need to continue the trend of getting the Semantic Web out of the research labs and back on the Web, where history has demonstrated that given a sufficient spark, truly extraordinary things can happen for often ordinary reasons—finding such reasons might be our greatest challenge yet.

[13] See `http://www.opendataday.org/`; retr. 2010/12/28

"If we wait for the moment when everything, absolutely everything is ready, we shall never begin."

—Ivan Turgenev

Bibliography

Adida, B. and Birbeck, M. (2008). RDFa Primer. W3C Working Group Note. `http://www.w3.org/TR/xhtml-rdfa-primer/`.

Alani, H., Brewster, C., and Shadbolt, N. (2006). Ranking ontologies with AKTiveRank. In *5th International Semantic Web Conference*, pages 1–15.

Alani, H., Dasmahapatra, S., O'Hara, K., and Shadbolt, N. (2003). Identifying communities of practice through ontology network analysis. *IEEE Intelligent Systems*, 18(2):18–25.

Allemang, D. and Hendler, J. A. (2008). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann/Elsevier.

Alvestrand, H. (2001). Tags for the Identification of Languages. RFC 3066. `http://www.ietf.org/rfc/rfc3066.txt`.

Anklesaria, F., McCahill, M., Lindner, P., Johnson, D., Torrey, D., and Alberti, B. (1993). The Internet Gopher Protocol (a distributed document search and retrieval protocol). RFC 1436. `http://tools.ietf.org/html/rfc1436`.

Anyanwu, K., Maduko, A., and Sheth, A. (2005). SemRank: ranking complex relationship search results on the semantic web. In *14th International Conference on World Wide Web*, pages 117–127.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F. (2002). *The Description Logic Handbook: Theory, Implementation and Application*. Cambridge University Press.

Balmin, A., Hristidis, V., and Papakonstantinou, Y. (2004). Objectrank: authority-based keyword search in databases. In *Proceedings of the 13th International Conference on Very Large Data Bases*, pages 564–575.

Barabási, A. L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286:509–512.

Batsakis, S., Petrakis, E. G. M., and Milios, E. (2009). Improving the performance of focused web crawlers. *Data Knowl. Eng.*, 68(10):1001–1013.

Bechhofer, S. and Volz, R. (2004). Patching Syntax in OWL Ontologies. In *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 668–682. Springer.

Beckett, D. (2010). RDF Syntaxes 2.0. In *W3C Workshop on RDF Next Steps*, Stanford, Palo Alto, CA, USA.

Beckett, D. and Berners-Lee, T. (2008). Turtle – Terse RDF Triple Language. W3C Team Submission. `http://www.w3.org/TeamSubmission/turtle/`.

Beckett, D. and McBride, B. (2004). RDF/XML Syntax Specification (Revised). W3C Recommendation. http://www.w3.org/TR/rdf-syntax-grammar/.

Belnap, N. (1977). A Useful Four-Valued Logic. *Modern Uses of Multiple-Valued Logic*, pages 5–37.

Berners-Lee, T. (1980). The ENQUIRE System – Short Description (1.1). Technical report, European Organisation for Nuclear Research. Available at http://www.w3.org/History/1980/Enquire/manual/; ed. Sean B. Palmer; retr. 2010/10/26.

Berners-Lee, T. (1993). A Brief History of the Web. W3C Design Issues. From http://www.w3.org/DesignIssues/TimBook-old/History.html; retr. 2010/10/27.

Berners-Lee, T. (1998a). Notation 3 – Ideas about Web architecture. W3C Design Issues. http://www.w3.org/DesignIssues/Notation3.html.

Berners-Lee, T. (1998b). Semantic Web Road map. http://www.w3.org/DesignIssues/Semantic.html.

Berners-Lee, T. (2006). Linked Data. W3C Design Issues. From http://www.w3.org/DesignIssues/LinkedData.html; retr. 2010/10/27.

Berners-Lee, T. (2010). The Future of RDF. W3C Design Issues. From http://www.w3.org/DesignIssues/RDF-Future.html; retr. 2010/10/28.

Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., and Sheets, D. (2006). Tabulator: Exploring and Analyzing linked data on the Semantic Web. In *The 3rd International Semantic Web User Interaction Workshop (SWUI06) workshop*.

Berners-Lee, T., Fielding, R. T., and Masinter, L. (2005). Uniform Resource Identifier (URI): Generic Syntax. RFC 3986. http://tools.ietf.org/html/rfc3986.

Berners-Lee, T. and Fischetti, M. (1999). *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5):34–43.

Bernstein, P. A., Melnik, S., and Mork, P. (2005). Interactive Schema Translation with Instance-Level Mappings. In *VLDB*, pages 1283–1286.

Birbeck, M. and McCarron, S. (2009). CURIE Syntax 1.0 – A syntax for expressing Compact URIs. W3C Recommendation. http://www.w3.org/TR/curie/.

Biron, P. V. and Malhotra, A. (2004). XML Schema Part 2: Datatypes Second Edition. W3C Recommendation. http://www.w3.org/TR/xmlschema-2/.

Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., and Velkov, R. (2011). OWLIM: A family of scalable semantic repositories. *Semantic Web Journal*. In press; available at http://www.semantic-web-journal.net/sites/default/files/swj97_0.pdf.

Bistarelli, S., Martinelli, F., and Santini, F. (2008). Weighted datalog and levels of trust. In *Proceedings of the The Third International Conference on Availability, Reliability and Security, ARES 2008, March 4-7, 2008, Technical University of Catalonia, Barcelona, Spain*, pages 1128–1134.

Bizer, C., Cyganiak, R., and Heath, T. (2008). How to Publish Linked Data on the Web. linkeddata.org Tutorial. http://linkeddata.org/docs/how-to-publish.

Bizer, C., Heath, T., and Berners-Lee, T. (2009a). Linked Data - The Story So Far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009b). DBpedia - A crystallization point for the Web of Data. *J. Web Sem.*, 7(3):154–165.

Bizer, C. and Schultz, A. (2009). The Berlin SPARQL Benchmark. *Int. J. Semantic Web Inf. Syst.*, 5(2):1–24.

Bizer, C. and Schultz, A. (2010). The R2R Framework : Publishing and Discovering Mappings on the Web. In *Consuming Linked Data (COLD) Workshop*.

Boldi, P., Codenotti, B., Santini, M., Vigna, S., and Vigna, S. (2002). UbiCrawler: a scalable fully distributed web crawler. *Software: Practice and Experience*, 34:2004.

Boley, H., Hallmark, G., Kifer, M., Paschke, A., Polleres, A., and Reynolds, D. (2010). RIF Core Dialect. W3C Recommendation. `http://www.w3.org/TR/rif-core/`.

Bonatti, P. A., Hogan, A., Polleres, A., and Sauro, L. (2011). Robust and Scalable Linked Data Reasoning Incorporating Provenance and Trust Annotations. *Journal of Web Semantics (**Provisionally Accepted**)*. `http://sw.deri.org/~aidanh/docs/saor_ann_jws_si.pdf`.

Bouquet, P., Stoermer, H., Mancioppi, M., and Giacomuzzi, D. (2006). OkkaM: Towards a Solution to the "Identity Crisis" on the Semantic Web. In *Proceedings of SWAP 2006, the 3rd Italian Semantic Web Workshop*, volume 201 of *CEUR Workshop Proceedings*.

Breslin, J. G., Passant, A., and Decker, S. (2009). *The Social Semantic Web*. Springer.

Brickley, D. and Guha, R. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. `http://www.w3.org/TR/rdf-schema/`.

Brickley, D., Guha, R., and Layman, A. (1998). Resource Description Framework (RDF) Schemas. W3C Working Draft. `http://www.w3.org/TR/1998/WD-rdf-schema-19980409/`.

Brin, S. and Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks*, 30(1-7):107–117.

Bush, V. (1945). As We May Think. *Atlantic Magazine*. Available at `http://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/3881/`; retr. 2010/10/27.

Calì, A., Gottlob, G., Lukasiewicz, T., Marnette, B., and Pieris, A. (2010). Datalog±: A Family of Logical Knowledge Representation and Query Languages for New Applications. In *LICS*, pages 228–242.

Castano, S., Ferrara, A., Montanelli, S., and Lorusso, D. (2008). Instance Matching for Ontology Population. In *SEBD*, pages 121–132.

Castillo, R. and Leser, U. (2010). Selecting Materialized Views for RDF Data. In *ICWE Workshops*, pages 126–137.

Chakrabarti, S., van den Berg, M., and Dom, B. (1999). Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. *Computer Networks*, 31(11-16):1623–1640.

Chen, Z., Kalashnikov, D. V., and Mehrotra, S. (2005). Exploiting relationships for object consolidation. In *IQIS '05: Proceedings of the 2nd international workshop on Information quality in information systems*, pages 47–58, New York, NY, USA. ACM Press.

Cheng, G., Ge, W., and Qu, Y. (2008a). Falcons: searching and browsing entities on the semantic web. In *World Wide Web*, pages 1101–1102.

Cheng, G., Ge, W., Wu, H., and Qu, Y. (2008b). Searching Semantic Web Objects Based on Class Hierarchies. In *Proceedings of Linked Data on the Web Workshop*.

Cheng, G. and Qu, Y. (2008). Term Dependence on the Semantic Web. In *International Semantic Web Conference*, pages 665–680.

Cheng, G. and Qu, Y. (2009). Searching Linked Objects with Falcons: Approach, Implementation and Evaluation. *Int. J. Semantic Web Inf. Syst.*, 5(3):49–70.

Clauset, A., Shalizi, C. R., and Newman, M. E. J. (2009). Power-Law Distributions in Empirical Data. *SIAM Rev.*, 51:661–703.

Cock, M. D. and Kerre, E. E. (2003). On (un)suitable fuzzy relations to model approximate equality. *Fuzzy Sets and Systems*, 133(2):137–153.

Cosmadakis, S. S., Gaifman, H., Kanellakis, P. C., and Vardi, M. Y. (1988). Decidable Optimization Problems for Database Logic Programs (Preliminary Report). In *STOC*, pages 477–490.

Cudré-Mauroux, P., Haghani, P., Jost, M., Aberer, K., and de Meer, H. (2009). idMesh: Graph-Based Disambiguation of Linked Data. In *WWW*, pages 591–600.

d'Aquin, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Sabou, M., and Motta, E. (2007). Characterizing Knowledge on the Semantic Web with Watson. In *5th International Workshop on Evaluation of Ontologies and Ontology-based Tools*, pages 1–10.

d'Aquin, M., Sabou, M., Motta, E., Angeletou, S., Gridinoc, L., Lopez, V., and Zablith, F. (2008). What Can be Done with the Semantic Web? An Overview Watson-based Applications. In *5th Workshop on Semantic Web Applications and Perspectives*.

de Bruijn, J., Eiter, T., Polleres, A., and Tompits, H. (2006). On Representational Issues About Combinations of Classical Theories with Nonmonotonic Rules. In *KSEM*, pages 1–22.

de Bruijn, J. and Heymans, S. (2007). Logical foundations of (e)rdf(s): Complexity and reasoning. In *ISWC/ASWC*, pages 86–99.

de Bruijn, J., Lara, R., Polleres, A., and Fensel, D. (2005a). OWL DL vs. OWL flight: conceptual modeling and reasoning for the semantic Web. In *International Conference on World Wide Web*, pages 623–632.

de Bruijn, J., Polleres, A., Lara, R., and Fensel, D. (2005b). OWL$^-$. Final draft d20.1v0.2, WSML.

Dean, J. and Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*, pages 137–150.

Decker, S., Erdmann, M., Fensel, D., and Studer, R. (1998). Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In *DS-8: IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics*, pages 351–369, Deventer, The Netherlands, The Netherlands. Kluwer, B.V.

Decker, S., Melnik, S., van Harmelen, F., Fensel, D., Klein, M. C. A., Broekstra, J., Erdmann, M., and Horrocks, I. (2000). The Semantic Web: The Roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–74.

Delbru, R., Polleres, A., Tummarello, G., and Decker, S. (2008). Context Dependent Reasoning for Semantic Documents in Sindice. In *Proc. of 4th SSWS Workshop*.

Delbru, R., Toupikov, N., Catasta, M., and Tummarello, G. (2010a). A Node Indexing Scheme for Web Entity Retrieval. In *Proceedings of the Extended Semantic Web Conference (ESWC 2010)*.

Delbru, R., Toupikov, N., Catasta, M., Tummarello, G., and Decker, S. (2010b). Hierarchical Link Analysis for Ranking Web Data. In *ESWC (2)*, pages 225–239.

Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., and Gori, M. (2000). Focused Crawling Using Context Graphs. In *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 527–534, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ding, L. and Finin, T. (2006). Characterizing the Semantic Web on the Web. In *International Semantic Web Conference*, pages 242–257.

Ding, L., Finin, T. W., Joshi, A., Pan, R., Cost, R. S., Peng, Y., Reddivari, P., Doshi, V., and Sachs, J. (2004). Swoogle: a search and metadata engine for the semantic web. In *CIKM*, pages 652–659.

Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., and Kolari, P. (2005). Finding and ranking knowledge on the Semantic Web. In *4th International Semantic Web Conference*, pages 156–170.

Dong, H., Hussain, F. K., and Chang, E. (2009). State of the Art in Semantic Focused Crawlers. In *ICCSA '09: Proceedings of the International Conference on Computational Science and Its Applications*, pages 910–924, Berlin, Heidelberg. Springer-Verlag.

Ehrig, M. and Maedche, A. (2003). Ontology-focused crawling of Web documents. In *SAC '03: Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 1174–1178. ACM.

Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16.

Engelbart, D. (1962). Augmenting Himan Intellect: A Conceptual Framework. Summary Report AFOSR-3233. Available at `http://www.invisiblerevolution.net/engelbart/full_62_paper_augm_hum_int.html`; retr. 2010/10/27.

Engelbart, D. (1968). Study for the development of Human Augmentation Techniques. Final Report NAS1-5904. Available at `http://sloan.stanford.edu/MouseSite/Archive/Post68/FinalReport1968/study68index.html`; retr. 2010/10/27.

Erling, O. and Mikhailov, I. (2009). RDF Support in the Virtuoso DBMS. In *Networked Knowledge – Networked Media*, volume 221 of *Studies in Computational Intelligence*, pages 7–24. Springer.

Fan, W., Jia, X., Li, J., and Ma, S. (2009). Reasoning about Record Matching Rules. *PVLDB*, 2(1):407–418.

Feigenbaum, L. (2010). Cambridge Semantics Position. In *W3C Workshop on RDF Next Steps*, Stanford, Palo Alto, CA, USA.

Fensel, D. and van Harmelen, F. (2007). Unifying Reasoning and Search to Web Scale. *IEEE Internet Computing*, 11(2):96, 94–95.

Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D. L., and Patel-Schneider, P. F. (2001). OIL: An Ontology Infrastructure for the Semantic Web. *IEEE Intelligent Systems*, 16(2):38–45.

Fernández, J. D., Gutierrez, C., and Martínez-Prieto, M. A. (2010). RDF compression: basic approaches. In *WWW*, pages 1091–1092.

Ferrara, A., Lorusso, D., Stamou, G. B., Stoilos, G., Tzouvaras, V., and Venetis, T. (2008). Resolution of Conflicts Among Ontology Mappings: a Fuzzy Approach. In *OM*.

Fielding, R. T., Gettys, J., Mogul, J. C., Frystyk, H., Masinter, L., Leach, P. J., and Berners-Lee, T. (1999). Hypertext Transfer Protocol – HTTP/1.1. RFC 2616. `http://www.ietf.org/rfc/rfc2616.txt`.

Finin, T. W. and Joshi, A. (2002). Agents, Trust, and Information Access on the Semantic Web. *SIGMOD Record*, 31(4):30–35.

Flouris, G., Fundulaki, I., Pediaditis, P., Theoharis, Y., and Christophides, V. (2009). Coloring RDF Triples to Capture Provenance. In *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, pages 196–212.

Franz, T., Schultz, A., Sizov, S., and Staab, S. (2009). TripleRank: Ranking Semantic Web Data by Tensor Decomposition. In *International Semantic Web Conference*, pages 213–228.

Ghanem, T. M. and Aref, W. G. (2004). Databases Deepen the Web. *IEEE Computer*, 37(1):116–117.

Ghilardi, S., Lutz, C., and Wolter, F. (2006). 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.

Glaser, H., Jaffri, A., and Millard, I. C. (2009). Managing Co-reference on the Semantic Web. In *3rd International Workshop on Linked Data on the Web (LDOW2009)*.

Glaser, H., Millard, I., and Jaffri, A. (2008). `RKBExplorer.com`: A knowledge driven infrastructure for linked data providers. In *ESWC Demo*, Lecture Notes in Computer Science, pages 797–801. Springer.

Gleich, D., Zhukov, L., and Berkhin, P. (2004). Fast Parallel PageRank: A Linear System Approach. Technical Report YRL-2004-038, Yahoo! Research Labs. `http://www.stanford.edu/~dgleich/publications/prlinear-dgleich.pdf`.

Glimm, B. and Kazakov, Y. (2008). Role conjunctions in expressive description logics. In *LPAR*, pages 391–405.

Glimm, B. and Rudolph, S. (2010). Status QIO: Conjunctive Query Entailment Is Decidable. In *KR*.

Golbreich, C. and Wallace, E. K. (2009). OWL 2 Web Ontology Language: New Features and Rationale. W3C Recommendation. http://www.w3.org/TR/owl2-new-features/.

Grant, J. and Beckett, D. (2004). RDF Test Cases. W3C Recommendation. `http://www.w3.org/TR/rdf-testcases/`.

Grau, B. C. (2007). OWL 1.1 Web Ontology Language Tractable Fragments. WebOnt Editor's Draft. `http://www.webont.org/owl/1.1/tractable.html`.

Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P. F., and Sattler, U. (2008). OWL 2: The next step for OWL. *J. Web Sem.*, 6(4):309–322.

Grau, B. C., Motik, B., Wu, Z., Fokoue, A., and Lutz, C. (2009). OWL 2 Web Ontology Language: Profiles. W3C Recommendation. http://www.w3.org/TR/owl2-profiles/.

Grosof, B., Horrocks, I., Volz, R., and Decker, S. (2004). Description Logic Programs: Combining Logic Programs with Description Logic. In *13th International Conference on World Wide Web*.

Guo, Y., Pan, Z., and Heflin, J. (2005). LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3):158–182.

Haas, L. M., Hentschel, M., Kossmann, D., and Miller, R. J. (2009). Schema AND Data: A Holistic Approach to Mapping, Resolution and Fusion in Information Integration. In *ER*, pages 27–40.

Halevy, A. Y., Norvig, P., and Pereira, F. (2009). The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, 24(2):8–12.

Halpin, H., Hayes, P. J., McCusker, J. P., McGuinness, D. L., and Thompson, H. S. (2010a). When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data. In *International Semantic Web Conference (1)*, pages 305–320.

Halpin, H., Herman, I., and Hayes, P. (2010b). When owl:sameAs isn't the Same: An Analysis of Identity Links on the Semantic Web. In *Linked Data on the Web WWW2010 Workshop (LDOW2010)*.

Harth, A. (2009). VisiNav: Visual Web Data Search and Navigation. In *DEXA*, pages 214–228.

Harth, A. (2010). *Exploring Linked Data at Web Scale*. PhD thesis, Digital Enterprise Research Institute, National University of Ireland, Galway.

Harth, A. and Gassert, H. (2005). On Searching and Displaying RDF Data from the Web. In *ESWC demo session*.

Harth, A., Kinsella, S., and Decker, S. (2009). Using Naming Authority to Rank Data and Ontologies for Web Search. In *International Semantic Web Conference*, pages 277–292.

Harth, A., Umbrich, J., and Decker, S. (2006). MultiCrawler: A Pipelined Architecture for Crawling and Indexing Semantic Web Data. In *5th International Semantic Web Conference*, pages 258–271.

Harth, A., Umbrich, J., Hogan, A., and Decker, S. (2007). YARS2: A Federated Repository for Querying Graph Structured Data from the Web. In *6th International Semantic Web Conference, 2nd Asian Semantic Web Conference*, pages 211–224.

Hartig, O., Harth, A., and Sequeda, J., editors (2010). *Proceedings of the First International Workshop on Consuming Linked Data, Workshop at ISWC2010, Shanghai, China, November 8*, volume 665 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Hawke, S. (2010). RIF In RDF. W3C Working Draft. `http://www.w3.org/TR/rif-in-rdf/`.

Hayes, P. (2004). RDF Semantics. W3C Recommendation. http://www.w3.org/TR/rdf-mt/.

Heflin, J., Hendler, J., and Luke, S. (1999). SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078, Dept. of Computer Science, University of Maryland.

Hendler, J. and McGuinness, D. L. (2000). The DARPA Agent Markup Language. *IEEE Intelligent Systems*, 15(6):67–73.

Hendler, J. A. (2009). Tonight's Dessert: Semantic Web Layer Cakes. In *ESWC*, page 1.

Hepp, M. (2009). Product Variety, Consumer Preferences, and Web Technology: Can the Web of Data Reduce Price Competition and Increase Customer Satisfaction? In *EC-Web*, page 144.

Heydon, A. and Najork, M. (1999). Mercator: A Scalable, Extensible Web Crawler. *World Wide Web*, 2:219–229.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S. (2009). OWL 2 Web Ontology Language Primer. W3C Recommendation. http://www.w3.org/TR/owl2-primer/.

Hitzler, P. and van Harmelen, F. (2010). A Reasonable Semantic Web. *Semantic Web Journal – Interoperability, Usability, Applicability*, 1(1).

Hogan, A. and Decker, S. (2009). On the Ostensibly Silent 'W' in OWL 2 RL. In *Third International Conference on Web Reasoning and Rule Systems, (RR2009)*, pages 118–134.

Hogan, A., Harth, A., and Decker, S. (2006). ReConRank: A Scalable Ranking Method for Semantic Web Data with Context. In *2nd Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2006).*

Hogan, A., Harth, A., and Decker, S. (2007a). Performing Object Consolidation on the Semantic Web Data Graph. In *1st I3 Workshop: Identity, Identifiers, Identification Workshop.*

Hogan, A., Harth, A., Passant, A., Decker, S., and Polleres, A. (2010a). Weaving the Pedantic Web. In *3rd International Workshop on Linked Data on the Web (LDOW2010)*, Raleigh, USA.

Hogan, A., Harth, A., and Polleres, A. (2008). SAOR: Authoritative Reasoning for the Web. In *3rd Asian Semantic Web Conference (ASWC)*, volume 5367, pages 76–90, Bankok, Thailand.

Hogan, A., Harth, A., and Polleres, A. (2009a). Scalable Authoritative OWL Reasoning for the Web. *International Journal on Semantic Web and Information Systems*, 5(2).

Hogan, A., Harth, A., and Polleres, A. (2009b). Scalable Authoritative OWL Reasoning for the Web. *Int. J. Semantic Web Inf. Syst.*, 5(2).

Hogan, A., Harth, A., Umbrich, J., and Decker, S. (2007b). Towards a Scalable Search and Query Engine for the Web. In *16th International Conference on World Wide Web (poster proc.).*

Hogan, A., Harth, A., Umbrich, J., Kinsella, S., Polleres, A., and Decker, S. (2010b). Searching and Browsing Linked Data with SWSE: the Semantic Web Search Engine. Technical Report DERI-TR-2010-07-23, Digital Enterprise Research Institute, Galway. `http://www.deri.ie/fileadmin/documents/DERI-TR-2010-07-23.pdf` (**Under Review** for the *Journal of Web Semantics*).

Hogan, A., Pan, J. Z., Polleres, A., and Decker, S. (2010c). SAOR: Template Rule Optimisations for Distributed Reasoning over 1 Billion Linked Data Triples. In *International Semantic Web Conference.*

Hogan, A., Polleres, A., Umbrich, J., and Zimmermann, A. (2010d). Some entities are more equal than others: statistical methods to consolidate Linked Data. In *4th International Workshop on New Forms of Reasoning for the Semantic Web: Scalable and Dynamic (NeFoRS2010).*

Hogan, A., Zimmermann, A., Umbrich, J., Polleres, A., and Decker, S. (2010e). Scalable and Distributed Methods for Resolving, Consolidating, Matching and Disambiguating Entities in Linked Data Corpora. *Journal of Web Semantics (**Under Review**).* `http://sw.deri.org/~aidanh/docs/entcons_jws_2010.pdf`.

Horrocks, I. and Patel-Schneider, P. F. (2004). Reducing OWL entailment to description logic satisfiability. *Journal of Web Semamtics*, 1(4):345–357.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., and Dean, M. (2004). SWRL: A

Semantic Web Rule Language Combining OWL and RuleML. W3C Recommendation. http://www.w3.org/Submission/SWRL/.

Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. (2002). Reviewing the Design of DAML+OIL: An Ontology Language for the Semantic Web. In *AAAI/IAAI*, pages 792–797.

Hu, W., Chen, J., Cheng, G., and Qu, Y. (2010). ObjectCoref and Falcon-AO: Results for OAEI 2010. In *Fifth International Workshop on Ontology Matching*.

Huang, Z. and van Harmelen, F. (2008). Using Semantic Distances for Reasoning with Inconsistent Ontologies. In *International Semantic Web Conference*, pages 178–194.

Jacobs, I. and Walsh, N. (2004). Architecture of the World Wide Web, Volume One. http://www.w3.org/TR/webarch/.

Jentzsch, A., Zhao, J., Hassanzadeh, O., Cheung, K.-H., Samwald, M., and Andersson, B. (2009). Linking Open Drug Data. In *International Conference on Semantic Systems (I-SEMANTICS?09)*.

Jérôme Euzenat, P. S. (2007). *Ontology matching*. Springer.

Jiménez-Ruiz, E., Grau, B. C., Sattler, U., Schneider, T., and Llavori, R. B. (2008). 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)*.

Jones, N. D., Gomard, C. K., Sestoft, P., Andersen, L. O., and Mogensen, T. (1993). *Partial Evaluation and Automatic Program Generation*. Prentice Hall International.

Kalyanpur, A., Parsia, B., Sirin, E., and Grau, B. C. (2006). Repairing Unsatisfiable Concepts in OWL Ontologies. In *ESWC*, pages 170–184.

Kifer, M. and Subrahmanian, V. S. (1992). Theory of Generalized Annotated Logic Programming and its Applications. *J. Log. Program.*, 12(3&4).

Kiryakov, A., Ognyanoff, D., Velkov, R., Tashev, Z., and Peikov, I. (2009). LDSR: a Reason-able View to the Web of Linked Data. In *Semantic Web Challenge (ISWC2009)*.

Kiryakov, A., Ognyanov, D., and Manov, D. (2005). OWLIM - A Pragmatic Semantic Repository for OWL. In *Web Information Systems Engineering Workshops*, LNCS, pages 182–192, New York, USA.

Klawonn, F. (2003). Should fuzzy equality and similarity satisfy transitivity? comments on the paper by m. de cock and e. kerre. *Fuzzy Sets and Systems*, 133(2):175–180.

Kleinberg, J. M. (1999). Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632.

Klyne, G. and Carroll, J. J. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation. http://www.w3.org/TR/rdf-concepts/.

Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M., Bizer, C., and Lee, R. (2009). Media Meets Semantic Web - How the BBC Uses DBpedia and Linked Data to Make Connections. In *ESWC*, pages 723–737.

Kolovski, V., Wu, Z., and Eadon, G. (2010). Optimizing Enterprise-scale OWL 2 RL Reasoning in a Relational Database System. In *International Semantic Web Conference*.

Komorowski, H. J. (1982). 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.

Lassila, O. and Swick, R. R. (1999). Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation. `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/`.

Lee, H.-T., Leonard, D., Wang, X., and Loguinov, D. (2008). IRLbot: scaling to 6 billion pages and beyond. In *WWW*, pages 427–436.

Lei, Y., Uren, V., and Motta, E. (2006). Semsearch: A search engine for the semantic web. In *14th International Conference on Knowledge Engineering and Knowledge Management*, pages 238–245.

Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., and Savo, D. F. (2010). Inconsistency-Tolerant Semantics for Description Logics. In *RR*, pages 103–117.

Liu, B. and Hu, B. (2005). An evaluation of rdf storage systems for large data applications. In *SKG*, page 59.

Lloyd, J. W. (1987). *Foundations of Logic Programming (2nd edition)*. Springer-Verlag.

Lloyd, J. W. and Shepherdson, J. C. (1991). Partial Evaluation in Logic Programming. *J. Log. Program.*, 11(3&4):217–242.

Lopes, N., Polleres, A., Straccia, U., and Zimmermann, A. (2010a). AnQL: SPARQLing Up Annotated RDFS. In *The Semantic Web - ISWC 20010, 9th International Semantic Web Conference, ISWC 2010, Shangai, Cina, November 7-11, to appear. Proceedings.*

Lopes, N., Zimmermann, A., Hogan, A., Lukacsy, G., Polleres, A., Straccia, U., and Decker, S. (2010b). RDF Needs Annotations. In *W3C Workshop on RDF Next Steps*, Stanford, Palo Alto, CA, USA.

Lu, J., Ma, L., 0007, L. Z., Brunner, J.-S., Wang, C., Pan, Y., and Yu, Y. (2007). Sor: A practical system for ontology storage, reasoning and search. In *VLDB*, pages 1402–1405.

Luke, S., Spector, L., Rager, D., and Hendler, J. A. (1997). Ontology-based Web Agents. In *Agents*, pages 59–66.

Lutz, C., Walther, D., and Wolter, F. (2007). Conservative extensions in expressive description logics. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 453–458.

Ma, L., Yang, Y., Qiu, Z., Xie, G. T., Pan, Y., and Liu, S. (2006). Towards a Complete OWL Ontology Benchmark. In *ESWC*, pages 125–139.

Ma, Y. and Hitzler, P. (2009). Paraconsistent Reasoning for OWL 2. In *RR*, pages 197–211.

Maier, F. (2010). Extending Paraconsistent $\mathcal{SROIQ}$. In *RR*, pages 118–132.

Manola, F., Miller, E., and McBride, B. (2004). RDF Primer. W3C Recommendation. `http://www.w3.org/TR/rdf-primer/`.

Martin, M., Unbehauen, J., and Auer, S. (2010). Improving the Performance of Semantic Web Applications with SPARQL Query Caching. In *ESWC (2)*, pages 304–318.

McCusker, J. P. and McGuinness, D. L. (2010). Towards Identity in Linked Data. In *In Proc. of OWL: Experience and Directions, San Francisco, USA*.

McGuinness, D. L., Fikes, R., Hendler, J. A., and Stein, L. A. (2002). DAML+OIL: An Ontology Language for the Semantic Web. *IEEE Intelligent Systems*, 17(5):72–80.

McGuinness, D. L. and van Harmelen, F. (2004). OWL Web Ontology Language Overview. W3C Recommendation. `http://www.w3.org/TR/owl-features/`.

Michalowski, M., Thakkar, S., and Knoblock, C. A. (2003). Exploiting Secondary Sources for Automatic Object Consolidation. In *Proceeding of 2003 KDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*.

Michel, B. S., Nikoloudakis, K., Reiher, P. L., and Zhang, L. (2000). URL Forwarding and Compression in Adaptive Web Caching. In *INFOCOM*, pages 670–678.

Miles, A., Baker, T., and Swick, R. (2006). Best Practice Recipes for Publishing RDF Vocabularies. `http://www.w3.org/TR/2006/WD-swbp-vocab-pub-20060314/l`. Superceded by Berrueta and Phipps: `http://www.w3.org/TR/swbp-vocab-pub/`.

Monaghan, F. and O'Sullivan, D. (2007). Leveraging ontologies, context and social networks to automate photo annotation. In *SAMT*, pages 252–255.

Motik, B. (2004). *Web Ontology Reasoning with Logic Databases*. PhD thesis, AIFB, Karlsruhe, Germany.

Motik, B., Horrocks, I., and Sattler, U. (2009a). Bridging the gap between OWL and relational databases. *J. Web Sem.*, 7(2):74–89.

Motik, B., Patel-Schneider, P. F., and Grau, B. C. (2009b). OWL 2 Web Ontology Language Direct Semantics. W3C Recommendation. http://www.w3.org/TR/owl2-direct-semantics/.

Motik, B., Patel-Schneider, P. F., and Parsia, B. (2009c). OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C Recommendation. http://www.w3.org/TR/owl2-syntax/.

Motta, E. (2000). Enabling Knowledge Creation, Sharing and Reuse on the World-Wide-Web. In *AIMSA*, pages 358–361.

Muñoz, S., Pérez, J., and Gutiérrez, C. (2007). Minimal Deductive Systems for RDF. In *ESWC*, pages 53–67.

Muñoz, S., Pérez, J., and Gutierrez, C. (2009). Simple and Efficient Minimal RDFS. *J. Web Sem.*, 7(3):220–234.

Najork, M. and Wiener, J. L. (2001). Breadth-First Search Crawling Yields High-Quality Pages. In *In Proc. 10th International World Wide Web Conference*, pages 114–118.

Nelson, T. H. (1965). Complex information processing: a file structure for the complex, the changing and the indeterminate. In *ACM Annual Conference/Annual Meeting*.

Neumann, T. and Weikum, G. (2010). The RDF-3X engine for scalable management of RDF data. *VLDB J.*, 19(1):91–113.

Newcombe, H. B., Kennedy, J. M., Axford, S. J., and James, A. P. (1959). Automatic Linkage of Vital Records: Computers can be used to extract "follow-up" statistics of families from files of routine records. *Science*, 130:954–959.

Nikolov, A., Uren, V. S., Motta, E., and Roeck, A. N. D. (2008). Integration of Semantically Annotated Data by the KnoFuss Architecture. In *EKAW*, pages 265–274.

Noessner, J., Niepert, M., Meilicke, C., and Stuckenschmidt, H. (2010). Leveraging terminological structure for object reconciliation. In *ESWC (2)*, pages 334–348.

Olson, M. A., Bostic, K., and Seltzer, M. I. (1999). Berkeley DB. In *USENIX Annual Technical Conference, FREENIX Track*, pages 183–191.

O'Reilly, T. (2006). *Web 2.0 Principles and Best Practices.* O'Reilly Media.

Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., and Tummarello, G. (2008). Sindice.com: a document-oriented lookup index for open linked data. *IJMSO*, 3(1):37–52.

Oren, E., Kotoulas, S., Anadiotis, G., Siebes, R., ten Teije, A., and van Harmelen, F. (2009a). Marvin: Distributed reasoning over large-scale Semantic Web data. *J. Web Sem.*, 7(4):305–316.

Oren, E., Kotoulas, S., Anadiotis, G., Siebes, R., ten Teije, A., and van Harmelen, F. (2009b). Marvin: Distributed reasoning over large-scale Semantic Web data. *J. Web Sem.*, 7(4):305–316.

Oren, E. and Tummarello, G. (2007). A Lookup Index for Semantic Web Resources. In *Workshop on Scripting for the Semantic Web (SFSW).*

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project.

Pant, G. and Srinivasan, P. (2005). Learning to crawl: Comparing classification schemes. *ACM Trans. Inf. Syst.*, 23(4):430–462.

Pietriga, E., Bizer, C., Karger, D. R., and Lee, R. (2006). Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In *International Semantic Web Conference*, pages 158–171.

Polleres, A. (2007). From SPARQL to rules (and back). In *WWW*, pages 787–796.

Polleres, A., Feier, C., and Harth, A. (2006). Rules with Contextually Scoped Negation. In *ESWC*, pages 332–347.

Polleres, A., Hogan, A., and Harth, A. (2010). Can we ever catch up with the Web? *Semantic Web Journal – Interoperability, Usability, Applicability*, 1(1).

Popitsch, N. and Haslhofer, B. (2010). DSNotify: handling broken links in the web of data. In *WWW*, pages 761–770.

Prud'hommeaux, E. and Seaborne, A. (2008). SPARQL Query Language for RDF. W3C Recommendation. `http://www.w3.org/TR/rdf-sparql-query/`.

Raimond, Y., Sutton, C., and Sandler, M. B. (2009). Interlinking Music-Related Data on the Web. *IEEE MultiMedia*, 16(2):52–63.

Ramakrishnan, R., Srivastava, D., and Sudarshan, S. (1990). Rule Ordering in Bottom-Up Fixpoint Evaluation of Logic Programs. In *Proc. of 16th VLDB*, pages 359–371.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95.

Sabou, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Motta, E., d'Aquin, M., and Dzbor, M. (2007). WATSON: A Gateway for the Semantic Web. In *ESWC 2007 poster session.*

Salvadores, M., Correndo, G., Rodriguez-Castro, B., Gibbins, N., Darlington, J., and Shadbolt, N. R. (2009). Linksb2n: Automatic data integration for the semantic web. In *OTM Conferences (2)*, pages 1121–1138.

Scharffe, F., Liu, Y., and Zhou, C. (2009). RDF-AI: an Architecture for RDF Datasets Matching, Fusion and Interlink. In *IJCAI 2009 Workshop on Identity, Reference, and Knowledge Representation (IR-KR)*.

Schenk, S., Saathoff, C., Staab, S., and Scherp, A. (2009). SemaPlorer - Interactive semantic exploration of data and media based on a federated cloud infrastructure. *J. Web Sem.*, 7(4):298–304.

Schenk, S. and Staab, S. (2008). Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the web. In *WWW*, pages 585–594.

Schlobach, S., Huang, Z., Cornet, R., and van Harmelen, F. (2007). Debugging Incoherent Terminologies. *J. Autom. Reasoning*, 39(3):317–349.

Schmidt, M., Hornung, T., Lausen, G., and Pinkel, C. (2009a). SP^{2}Bench: A SPARQL Performance Benchmark. In *ICDE*, pages 222–233.

Schmidt, M., Hornung, T., Lausen, G., and Pinkel, C. (2009b). SP^{2}Bench: A SPARQL Performance Benchmark. In *ICDE*, pages 222–233.

Schmidt-Schauß, M. and Smolka, G. (1991). Attributive Concept Descriptions with Complements. *Artif. Intell.*, 48(1):1–26.

Schneider, M. (2009). OWL 2 Web Ontology Language RDF-Based Semantics. W3C Recommendation. http://www.w3.org/TR/owl2-rdf-based-semantics/.

Shi, L., Berrueta, D., Fernández, S., Polo, L., and Fernández, S. (2008). Smushing RDF instances: are Alice and Bob the same open source developer? In *PICKME Workshop*.

Singh, S., Wick, M. L., and McCallum, A. (2010). Distantly Labeling Data for Large Scale Cross-Document Coreference. *CoRR*, abs/1005.4298.

Sintek, M. and Decker, S. (2002). TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In *1st International Semantic Web Conference*, pages 364–378.

Sleeman, J. and Finin, T. (2010). Learning Co-reference Relations for FOAF Instances. In *Poster and Demo Session at ISWC*.

Smith, M. K., Welty, C., and McGuinness, D. L. (2004). OWL Web Ontology Language Guide. W3C Recommendation. `http://www.w3.org/TR/owl-guide/`.

Staab, S., Angele, J., Decker, S., Erdmann, M., Hotho, A., Maedche, A., Schnurr, H.-P., Studer, R., and Sure, Y. (2000). Ai for the web - ontology-based community web portals. In *AAAI/IAAI*, pages 1034–1039.

Stocker, M., Seaborne, A., Bernstein, A., Kiefer, C., and Reynolds, D. (2008). SPARQL basic graph pattern optimization using selectivity estimation. In *WWW*, pages 595–604.

Stoer, M. and Wagner, F. (1997). A simple min-cut algorithm. *J. ACM*, 44(4):585–591.

Stonebraker, M. (1986). The Case for Shared Nothing. *IEEE Database Eng. Bull.*, 9(1):4–9.

Straccia, U. (2010). SoftFacts: A Top-k Retrieval Engine for Ontology Mediated Access to Relational Databases. In *SMC*, pages 4115–4122.

Stuckenschmidt, H. (2008). Debugging owl ontologies - a reality check. In *EON*.

Stuckenschmidt, H. (2009). A Semantic Similarity Measure for Ontology-Based Information. In *FQAS '09: Proceedings of the 8th International Conference on Flexible Query Answering Systems*, pages 406–417, Berlin, Heidelberg. Springer-Verlag.

ter Horst, H. J. (2005a). Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity. In *4th International Semantic Web Conference*, pages 668–684.

ter Horst, H. J. (2005b). Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3:79–115.

Thelwall, M. and Stuart, D. (2006). Web crawling ethics revisited: Cost, privacy, and denial of service. *Journal of the American Society for Information Science and Technology*, 57:1771–1779.

Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., and Decker, S. (2009). Sig.ma: Live views on the Web of Data. In *Semantic Web Challenge (ISWC2009)*.

Tummarello, G., Delbru, R., and Oren, E. (2007). Sindice.com: Weaving the Open Linked Data. In *ISWC/ASWC*, pages 552–565.

Ullman, J. D. (1989). *Principles of Database and Knowledge Base Systems*. Computer Science Press.

Umbrich, J., Harth, A., Hogan, A., and Decker, S. (2008). Four heuristics to guide structured content crawling. In *Proceedings of the 2008 Eighth International Conference on Web Engineering-Volume 00*, pages 196–202. IEEE Computer Society.

Umbrich, J., Villazón-Terrazas, B., and Hausenblas, M. (2010). Dataset Dynamics Compendium: A Comparative Study. In *Consuming Linked Data (COLD) Workshop*.

Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., and Bal, H. E. (2010). OWL Reasoning with WebPIE: Calculating the Closure of 100 Billion Triples. In *ESWC (1)*, pages 213–227.

Urbani, J., Kotoulas, S., Oren, E., and van Harmelen, F. (2009). Scalable Distributed Reasoning Using MapReduce. In *International Semantic Web Conference*, pages 634–649.

van Harmelen, F. and Fensel, D. (1999). Practical Knowledge Representation for the Web. In *Intelligent Information Integration*.

Vanderbei, R. J. (2008). *Linear Programming: Foundations and Extensions, 3rd ed.* Springer Verlag.

Volz, J., Bizer, C., Gaedke, M., and Kobilarov, G. (2009). Discovering and Maintaining Links on the Web of Data. In *International Semantic Web Conference*, pages 650–665.

Vrandečíc, D., Krötzsch, M., Rudolph, S., and Lösch, U. (2010). Leveraging Non-Lexical Knowledge for the Linked Open Data Web. *Review of April Fool's day Transactions (RAFT)*, 5:18–27.

Wagner, G. (2003). Web rules need two kinds of negation. In *PPSWR*, pages 33–50.

Wang, S.-Y., Guo, Y., Qasem, A., and Heflin, J. (2005). Rapid Benchmarking for Semantic Web Knowledge Base Systems. In *International Semantic Web Conference*, pages 758–772.

Wang, T. D., Parsia, B., and Hendler, J. A. (2006a). A Survey of the Web Ontology Landscape. In *International Semantic Web Conference*, pages 682–694.

Wang, T. D., Parsia, B., and Hendler, J. A. (2006b). A Survey of the Web Ontology Landscape. In

Proceedings of the 5th International Semantic Web Conference (ISWC 2006), pages 682–694, Athens, GA, USA.

Weaver, J. and Hendler, J. A. (2009). Parallel Materialization of the Finite RDFS Closure for Hundreds of Millions of Triples. In *International Semantic Web Conference (ISWC2009)*, pages 682–697.

Wood, D., Decker, S., and Herman, I., editors (2010). *Proceedings of the W3C Workshop – RDF Next Steps, Stanford, Palo Alto, CA, USA, June 26-27*. Online at `http://www.w3.org/2009/12/rdf-ws/`.

Wu, C. H., Apweiler, R., Bairoch, A., Natale, D. A., Barker, W. C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., Martin, M. J., Mazumder, R., O'Donovan, C., Redaschi, N., and Suzek, B. E. (2006). The Universal Protein Resource (UniProt): an expanding universe of protein information. *Nucleic Acids Research*, 34(Database-Issue):187–191.

Wu, Z., Eadon, G., Das, S., Chong, E. I., Kolovski, V., Annamalai, M., and Srinivasan, J. (2008). Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle. In *ICDE*, pages 1239–1248.

Yardeni, E. and Shapiro, E. Y. (1991). A Type System for Logic Programs. *J. Log. Program.*, 10(1/2/3&4):125–153.

Zadeh, L. A., Klir, G. J., and Yuan, B. (1996). *Fuzzy Sets, Fuzzy Logic, Fuzzy Systems*. World Scientific Press.

Zhang, X., Xiao, G., and Lin, Z. (2009). A Tableau Algorithm for Handling Inconsistency in OWL. In *ESWC*, pages 399–413.

Zhou, J., Ma, L., Liu, Q., 0007, L. Z., Yu, Y., and Pan, Y. (2006). Minerva: A scalable owl ontology storage and inference system. In *ASWC*, pages 429–443.

Appendix A

Prefixes

Herein, we enumerate the prefixes used throughout this thesis to abbreviate URIs. In Table A.1, we list the prefixes used for instance URIs; in Table A.2, we list the prefixes used for class or property URIs.

Prefix	**URI**
avtimbl:	http://www.advogato.org/person/timbl/foaf.rdf#
bmpersons:	http://www4.wiwiss.fu-berlin.de/bookmashup/persons/
dav:	http://www.openlinksw.com/dataspace/organization/dav#
dblpperson:	http://www4.wiwiss.fu-berlin.de/dblp/resource/person/
dbpedia:	http://dbpedia.org/resource/
enwiki:	http://en.wikipedia.org/wiki/
ex*:	*arbitrary example namespace*
eswc2006p:	http://www.eswc2006.org/people/#
fb:	http://rdf.freebase.com/ns/
identicauser:	http://identi.ca/user/45563
kingdoms:	http://lod.geospecies.org/kingdoms/
macs:	http://stitch.cs.vu.nl/alignments/macs/
sw:	http://data.semanticweb.org/id/
swid:	http://semanticweb.org/id/
sworg:	http://data.semanticweb.org/organization/
swpaper1:	http://data.semanticweb.org/conference/iswc/2006/paper-
swpaper2:	http://data.semanticweb.org/conference/iswc-aswc/2007/tracks/research/papers/
swpaper3:	http://data.semanticweb.org/conference/iswc/2009/paper/research/
swperson:	http://data.semanticweb.org/person/
vperson:	http://virtuoso.openlinksw.com/dataspace/person/
wikier:	http://www.wikier.org/foaf.rdf#

Table A.1: Used “data” prefixes

Prefix	URI
aifb:	http://www.aifb.kit.edu/id/
atomowl:	http://bblfish.net/work/atom-owl/2006-06-06/#
b2r:	http://bio2rdf.org/bio2rdf:
b2r2008:	http://bio2rdf.org/bio2rdf-2008.owl#
b2rns:	http://bio2rdf.org/ns
b2rr:	http://bio2rdf.org/bio2rdf_resource:
bill:	http://www.rdfabout.com/rdf/schema/usbill/
cc:	http://creativecommons.org/ns#
contact:	http://www.w3.org/2000/10/swap/pim/contact#
dbo:	http://dbpedia.org/ontology/
dbp:	http://dbpedia.org/property/
dbtropes:	http://dbtropes.org/resource/Main/
dc:	http://purl.org/dc/elements/1.1/
dcmit:	http://purl.org/dc/dcmitype/
dct:	http://purl.org/dc/terms/
doap:	http://usefulinc.com/ns/doap#
drugbank:	http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/
ecs:	http://rdf.ecs.soton.ac.uk/ontology/ecs#
esns:	http://ontologycentral.com/2009/01/eurostat/ns#
estoc:	http://ontologycentral.com/2009/01/eurostat/table_of_contents#
ex*:	*arbitrary example namespace*
factbook:	http://www4.wiwiss.fu-berlin.de/factbook/ns#
fb:	http://rdf.freebase.com/ns/
foaf:	http://xmlns.com/foaf/0.1/
frbr:	http://purl.org/vocab/frbr/core#
geonames:	http://www.geonames.org/ontology#
geospecies:	http://rdf.geospecies.org/ont/geospecies#
gr:	http://purl.org/goodrelations/v1#
kwa:	http://knowledgeweb.semanticweb.org/heterogeneity/alignment#
like:	http://ontologi.es/like#
lldegene:	http://linkedlifedata.com/resource/entrezgene/
lldlifeskim:	http://linkedlifedata.com/resource/lifeskim/
lldpubmed:	http://linkedlifedata.com/resource/pubmed/
loc:	http://sw.deri.org/2006/07/location/loc#
ludopinions:	http://skipforward.net/skipforward/resource/ludopinions/
mo:	http://purl.org/ontology/mo/
movie:	http://data.linkedmdb.org/resource/movie/
mu:	http://www.kanzaki.com/ns/music#
mvcb:	http://webns.net/mvcb/
opencyc:	http://sw.opencyc.org/2008/06/10/concept/
opiumfield:	http://rdf.opiumfield.com/lastfm/spec#
oplweb:	http://www.openlinksw.com/schemas/oplweb#
opwn:	http://www.ontologyportal.org/WordNet.owl#
owl:	http://www.w3.org/2002/07/owl#
own16:	http://www.ontologydesignpatterns.org.it/ont/own/own16.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#
ptime:	http://pervasive.semanticweb.org/ont/2004/06/time#
quaffing:	http://purl.org/net/schemas/quaffing/
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#
rev:	http://purl.org/stuff/rev#
rss:	http://purl.org/rss/1.0/
sc:	http://umbel.org/umbel/sc/
skipinions:	http://skipforward.net/skipforward/resource/seeder/skipinions/
skos:	http://www.w3.org/2004/02/skos/core#
sumo:	http://www.ontologyportal.org/SUMO.owl#
swrc:	http://swrc.ontoware.org/ontology#
uniprot:	http://purl.uniprot.org/core/
vote:	http://www.rdfabout.com/rdf/schema/vote/
wgs84:	http://www.w3.org/2003/01/geo/wgs84_pos#
wn:	http://xmlns.com/wordnet/1.6/
wrcc:	http://web.resource.org/cc
xfn:	http://vocab.sindice.com/xfn#
yago:	http://dbpedia.org/class/yago/
yagor:	http://www.mpii.de/yago/resource/

Table A.2: Used “vocabulary” prefixes

Appendix B

Rule Tables

This appendix lists Turtle-syntax rules for OWL 2 RL/RDF (§ B.1); for reference, we also include rules for RDF(S) (§ B.2) and pD* (§ B.3). Note that we highlight all authoritative variables (§ 5.4.2) in bold for all rulesets, and that we italicise the labels of OWL 2 RL/RDF rules which use features new to OWL 2.

B.1 OWL 2 RL/RDF Rules

Herein, we list the rule tables categorising supported OWL 2 RL/RDF rules [Grau et al., 2009] according to terminological and assertional arity of atoms in the body.

We begin with the rules supported in Chapter 5; these include:

- rules with no set body, which are roughly equivalent to (sometimes infinite) axiomatic triples (Table B.1);
- rules with only terminological atoms (Table B.2);
- rules with no terminological atoms and one assertional atom (Table B.3);
- and rules with terminological atom(s) and one assertional atom (Table B.4).

We deliberately do not support crossed-out rules as per discussion in Chapter 5. Further, as mentioned in Chapter 5, Table B.5 gives an indication as to how the least model of the assertional program can be complete even if the inferences from rules in Table B.2 (rules with only T-atoms) are omitted from the T-Box.

We then present all of the OWL 2 RL/RDF constraint rules—rules with the special `false` keyword as head—in Table B.6. These rules are used in Chapters 6 & 7 to detect noise.

In Table B.7, we list the rules used to support the semantics of equality, as required for consolidation presented in Chapter 7; Table B.8 lists the rules which can *directly* and *only* infer `owl:sameAs` relations. Note that in these rulesets, we denote rules that we omit by choice using double-cross-out, and thereafter, rules that we cannot currently support using single-cross out.

Finally, in Table B.9, we list the remaining OWL 2 RL/RDF rules which we currently do not support by any means.

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~~	~~$?l$ a $?dt$.~~	~~For all $?l$ in the value space of datatype $?dt$~~
~~dt-eq~~	~~$?l_1$ owl:sameAs $?l_2$.~~	~~For all $?l_1$ and $?l_2$ with the same data value~~
~~dt-diff~~	~~$?l_1$ owl:differentFrom $?l_2$.~~	~~For all $?l_1$ and $?l_2$ with different data values~~

Table B.1: Rules with empty body

TBody$(R) \neq \emptyset$, 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 $?c$; rdfs:subClassOf 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 B.2: Rules containing only T-atoms in the body

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

Table B.3: Rules with no T-atoms but precisely one A-atom in the body

TBody(R) $\neq \emptyset$ and $\lvert$ABody(R)$\rvert = 1$			
ID	**Body**		**Head**
	terminological	*assertional*	
prp-dom	$\boldsymbol{?p}$ rdfs:domain $?c$.	$?x\ ?p\ ?y$.	$?x$ a $?c$.
prp-rng	$\boldsymbol{?p}$ rdfs:range $?c$.	$?x\ ?p\ ?y$.	$?y$ a $?c$.
prp-symp	$\boldsymbol{?p}$ 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	$\boldsymbol{?c}$ 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 $\boldsymbol{?p}$.	$?u\ ?p\ ?v$.	$?u$ a $?x$.
cls-hv1	$\boldsymbol{?x}$ owl:hasValue $?y$; owl:onProperty $\boldsymbol{?p}$.	$?u$ a $?x$.	$?u\ ?p\ ?y$.
cls-hv2	$?x$ owl:hasValue $\boldsymbol{?y}$; owl:onProperty $\boldsymbol{?p}$.	$?u\ ?p\ ?y$.	$?u$ 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 B.4: Rules containing some T-atoms and precisely one A-atom in the body

ID	partially covered by recursive rule(s)
scm-cls	*incomplete* for `owl:Thing` membership inferences
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:* $\underline{\text{cls-svf1}}$, cax-sco
scm-svf2	*incomplete:* $\underline{\text{cls-svf1}}$, prp-spo1
scm-avf1	*incomplete:* $\underline{\text{cls-avf}}$, cax-sco
scm-avf2	*incomplete:* $\underline{\text{cls-avf}}$, prp-spo1
scm-int	cls-int2
scm-uni	cls-uni

Table B.5: Enumeration of the coverage of inferences in case of the omission of rules in Table B.2 wrt. inferencing over assertional knowledge by recursive application of rules in Table B.4: 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).

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 $?z_j$. ($i{\neq}j$)
eq-diff3	-	?x a owl:AllDifferent ; owl:distinctMembers ($?z_1$...$?z_n$) . $?z_i$ owl:sameAs $?z_j$. ($i{\neq}j$)
*eq-irp**	-	?x owl:differentFrom ?x .
prp-irp	**?p** a owl:IrreflexiveProperty .	?x ?p ?x .
prp-asyp	**?p** a owl:AsymmetricProperty	?x ?p ?y . ?y ?p ?x .
prp-pdw	$\boldsymbol{?p_1}$ owl:propertyDisjointWith $\boldsymbol{?p_2}$.	?x $?p_1$?y ; $?p_2$?y .
prp-adp	?x a owl:AllDisjointProperties . ?x owl:members ($\boldsymbol{?p_1}$...$\boldsymbol{?p_n}$) .	?u $?p_i$?y ; $?p_j$?y . ($i{\neq}j$)
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	$\boldsymbol{?c_1}$ owl:complementOf $\boldsymbol{?c_2}$.	?x a $?c_1$, $?c_2$.
cls-maxc1	**?x** owl:maxCardinality 0 . **?x** owl:onProperty **?p** .	?u a ?x ; ?p ?y .
cls-maxqc1	**?x** owl:maxQualifiedCardinality 0 . **?x** owl:onProperty **?p** . **?x** owl:onClass **?c** .	?u a ?x ; ?p ?y . ?y a ?c .
cls-maxqc2	**?x** owl:maxQualifiedCardinality 0 . **?x** owl:onProperty **?p** . **?x** owl:onClass owl:Thing .	?u a ?x ; ?p ?y .
cax-dw	$\boldsymbol{?c_1}$ owl:disjointWith $\boldsymbol{?c_2}$.	?x a $?c_1$, $?c_2$.
cax-adc	?x a owl:AllDisjointClasses . ?x owl:members ($\boldsymbol{?c_1}$...$\boldsymbol{?c_n}$) .	?z a $?c_i$, $?c_j$. ($i{\neq}j$)
dt-not-type*	-	?s ?p ?lt . [for ?lt an ill-typed literal]

Table B.6: Constraint Rules

ID	**Body**	**Head**
	assertional	
~~eq-ref~~	~~?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 .
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 . [?p $\neq$ rdf:type]	?s ?p ?o′ .

Table B.7: Rules that support the positive semantics of `owl:sameAs`

ID	**Body**		**Head**
	terminological	*assertional*	
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 ($\boldsymbol{?p_1}$ ... $\boldsymbol{?p_n}$)~~	~~?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** .~~ ~~**?x** owl:onClass **?c** .~~	~~?u a ?x .~~ ~~?u ?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 . **?x** owl:onProperty **?p** . **?x** owl:onClass owl:Thing .	?u a ?x . ?u ?p $?y_1$, $?y_2$.	$?y_1$ owl:sameAs $?y_2$

Table B.8: OWL 2 RL/RDF rules that directly produce `owl:sameAs` relations

ID	Body		Head
	terminological	*assertional*	
prp-trp	$\boldsymbol{?p}$ a owl:TransitiveProperty .	$?x\ ?p\ ?y$. $?y\ ?p\ ?z$.	$?x\ ?p\ ?z$
prp-spo2	$?p$ owl:propertyChainAxiom ($\boldsymbol{?p_1}$ ... $\boldsymbol{?p_n}$) .	$?u_1\ ?p_1\ ?u_2$. $?u_2\ ?p_2\ ?u_3$ . ... $?u_n\ ?p_n\ ?u_{n+1}$.	$?u_1\ ?p\ ?u_{n+1}$.
cls-int1	$?c$ owl:intersectionOf ($\boldsymbol{?c_1}$ ... $\boldsymbol{?c_n}$) .	$?y$ a $?c_1$, ... $?c_n$.	$?y$ a $?c$.
cls-svf1	$?x$ owl:someValuesFrom $\boldsymbol{?y}$; owl:onProperty $\boldsymbol{?p}$.	$?u\ ?p\ ?v$. $?v$ a $?y$.	$?u$ a $?x$.
cls-avf	$\boldsymbol{?x}$ owl:allValuesFrom $?y$; owl:onProperty $\boldsymbol{?p}$.	$?u\ ?p\ ?v$; a $?x$.	$?v$ a $?y$.

Table B.9: Remaining OWL 2 RL/RDF rules which we currently do not support

B.2 RDF(S) Rules

Herein, for reference, we list the various RDFS-related rulesets available at [Hayes, 2004]. We start with two simple entailment rules and the two variants thereof (lg/gl) used in RDF and RDFS (Table B.10). We then list the RDF axiomatic triples (Table B.11) and the RDF entailment rules (Table B.12), followed by the RDFS axiomatic triples (Table B.13) and the RDFS entailment rules (Table B.14). Finally, we also list the Extensional RDFS (eRDFS) entailment rules (Table B.15) and the Datatype (D) Entailment rules (Table B.16).

Note that for RDFS and pD*, we do not transcribe syntactic restrictions on what type of RDF constant can enter which position of a triple—such restrictions become moot when one considers (even intermediary) generalised triples.

ID	**Body**	**Head**
	assertional	
se1	*?u ?b ?x* .	*?u ?b* _:n .
se2	*?u ?b ?x* .	_:n *?b ?x* .
lg	*?u ?b ?l* .	*?u ?b* _:l . (_:l a surrogate for literal *?l*)
gl	*?u ?b* _:l .	*?u ?b ?l* . (_:l a surrogate for literal *?l*)

Table B.10: Simple entailment rules [Hayes, 2004]

rdf:type rdf:type rdf:Property .
rdf:subject rdf:type rdf:Property .
rdf:predicate rdf:type rdf:Property .
rdf:object rdf:type rdf:Property .
rdf:first rdf:type rdf:Property .
rdf:rest rdf:type rdf:Property .
rdf:subject rdf:type rdf:Property .
value rdf:type rdf:Property .
rdf:_1 rdf:type rdf:Property .
...
rdf:nil rdf:type rdf:List .

Table B.11: RDF axiomatic triples [Hayes, 2004]

ID	**Body**	**Head**
	assertional	
rdf1	*?u ?b ?y* .	*?b* a rdf:Property.
rdf2	*?u ?b ?l* .	_:l a rdf:XMLLiteral . (by lg for *?l* a valid rdf:XMLLiteral)
includes RDF axiomatic triples and simple-entailment rules lg/gl		

Table B.12: RDF entailment rules [Hayes, 2004]

rdf:type rdfs:domain rdfs:Resource .
rdfs:domain rdfs:domain rdf:Property .
rdfs:range rdfs:domain rdf:Property .
rdfs:subPropertyOf rdfs:domain rdf:Property .
rdfs:subClassOf rdfs:domain rdfs:Class .
rdf:subject rdfs:domain rdf:Statement .
rdf:predicate rdfs:domain rdf:Statement .
rdf:object rdfs:domain rdf:Statement .
rdfs:member rdfs:domain rdfs:Resource .
rdf:first rdfs:domain rdf:List .
rdf:rest rdfs:domain rdf:List .
rdfs:seeAlso rdfs:domain rdfs:Resource .
rdfs:isDefinedBy rdfs:domain rdfs:Resource .
rdfs:comment rdfs:domain rdfs:Resource .
rdfs:label rdfs:domain rdfs:Resource .
rdf:value rdfs:domain rdfs:Resource .
rdf:type rdfs:range rdfs:Class .
rdfs:domain rdfs:range rdfs:Class .
rdfs:range rdfs:range rdfs:Class .
rdfs:subPropertyOf rdfs:range rdf:Property .
rdfs:subClassOf rdfs:range rdfs:Class .
rdf:subject rdfs:range rdfs:Resource .
rdf:predicate rdfs:range rdfs:Resource .
rdf:object rdfs:range rdfs:Resource .
rdfs:member rdfs:range rdfs:Resource .
rdf:first rdfs:range rdfs:Resource .
rdf:rest rdfs:range rdf:List .
rdfs:seeAlso rdfs:range rdfs:Resource .
rdfs:isDefinedBy rdfs:range rdfs:Resource .
rdfs:comment rdfs:range rdfs:Literal .
rdfs:label rdfs:range rdfs:Literal .
rdfs:value rdfs:range rdfs:Resource .
rdf:Alt rdfs:subClassOf rdfs:Container .
rdf:Bag rdfs:subClassOf rdfs:Container .
rdf:Seq rdfs:subClassOf rdfs:Container .
rdfs:ContainerMembershipProperty rdfs:subClassOf rdf:Property .
rdfs:isDefinedBy rdfs:subPropertyOf rdfs:seeAlso .
rdf:XMLLiteral rdf:type rdfs:Datatype .
rdf:XMLLiteral rdfs:subClassOf rdfs:Literal .
rdfs:Datatype rdfs:subClassOf rdfs:Class .
rdf:_1 rdf:type rdfs:ContainerMembershipProperty .
rdf:_1 rdf:domain rdfs:Resource .
rdf:_1 rdfs:range rdfs:Resource .
rdf:_2 rdfs:type rdfs:ContainerMembershipProperty .
rdf:_2 rdfs:domain rdfs:Resource .
rdf:_2 rdfs:range rdfs:Resource .
...

Table B.13: RDFS axiomatic triples [Hayes, 2004]

ID	**Body**		**Head**
	terminological	*assertional*	
rdfs1		*?u ?b ?l* .	_:l a rdfs:Literal . (by lg for *?l* a plain literal)
rdfs2	***?v*** rdfs:domain *?x* .	*?u ?v ?y* .	*?u* a *?x* .
rdfs3	***?b*** rdfs:range *?x* .	*?u ?b ?v* .	*?v* a *?x* .
rdfs4a		*?u ?b ?x* .	*?u* a rdfs:Resource .
rdfs4b		*?u ?b ?v* .	*?v* a rdfs:Resource .
rdfs5	*?u* rdfs:subPropertyOf *?v* . *?v* rdfs:subPropertyOf *?x* .		*?u* rdfs:subPropertyOf *?x* .
rdfs6	*?u* a rdf:Property.		*?u* rdfs:subPropertyOf *?u* .
rdfs7	***?b*** rdfs:subPropertyOf *?c* .	*?u ?b ?y* .	*?u ?c ?y* .
rdfs8	*?u* a rdfs:Class.		*?u* rdfs:subClassOf rdfs:Resource.
rdfs9	***?u*** rdfs:subClassOf *?x* .	*?v* a *?u* .	*?v* a *?x* .
rdfs10	*?u* a rdfs:Class.		*?u* rdfs:subClassOf *?u* .
rdfs11	*?u* rdfs:subClassOf *?v* . *?v* rdfs:subClassOf *?x* .		*?u* rdfs:subClassOf *?x* .
rdfs12	*?u* a rdfs:ContainerMembershipProperty .		*?u* rdfs:subPropertyOf rdfs:member .
rdfs13	*?u* a rdfs:Datatype .		*?u* rdfs:subClassOf rdfs:Literal .
includes RDF(S) axiomatic triples, RDF-entailment rules and rules lg/gl			

Table B.14: RDFS entailment [Hayes, 2004]

ID	**Body**	**Head**
	terminological	
ext1	*?u* rdfs:domain *?v* . *?v* rdfs:subClassOf *?z* .	*?u* rdfs:domain *?z* .
ext2	*?u* rdfs:range *?v* . *?v* rdfs:subClassOf *?z* .	*?u* rdfs:range *?z* .
ext3	*?u* rdfs:domain *?v* . *?w* rdfs:subPropertyOf *?u* .	*?w* rdfs:domain *?v* .
ext4	*?u* rdfs:range *?v* . *?w* rdfs:subPropertyOf *?u* .	*?w* rdfs:range *?v* .
ext5	a rdfs:subPropertyOf *?w* . *?w* rdfs:domain *?v* .	rdfs:Resource rdfs:subClassOf *?v* .
ext6	rdfs:subClassOf rdfs:subPropertyOf *?w* . *?w* rdfs:domain *?v* .	rdfs:Class rdfs:subClassOf *?v* .
ext7	rdfs:subPropertyOf rdfs:subPropertyOf *?w* . *?w* rdfs:domain *?v* .	rdf:Property rdfs:subClassOf *?v* .
ext8	rdfs:subClassOf rdfs:subPropertyOf *?w* . *?w* rdfs:range *?v* .	rdfs:Class rdfs:subClassOf *?v* .
ext9	rdfs:subPropertyOf rdfs:subPropertyOf *?w* . *?w* rdfs:range *?v* .	rdf:Property rdfs:subClassOf *?v* .

Table B.15: Extensional RDFS entailment [Hayes, 2004]

ID	**Body**	**Head**
	assertional	
rdfD1	*?d* a rdfs:Datatype. *?u ?b ?l*^^*?d* .	_:l a *?d* . (by lg for *?l* a datatype literal)
rdfD2	*?d* a rdfs:Datatype. *?u ?b ?l*^^*?d*	*?u ?b ?m*^^*?d* . (where *?l*^^*?d* = *?m*^^*?d*)
rdfD3	*?d* a rdfs:Datatype. *?e* a rdfs:Datatype. *?u ?b ?l*^^*?d* .	*?u ?b ?m*^^*?e* . (where *?l*^^*?d* = *?m*^^*?e*)
xsd 1a	*?u ?b* “s” .	*?u ?b* “s”^^xsd:string .
xsd 1b	*?u ?b* “s”^^xsd:string .	*?u ?b* “s” .

Table B.16: Datatype entailment rules [Hayes, 2004]

B.3 pD* Rules

Finally, we list the additional P-axiomatic triples (Table B.17) and P-entailment rules (Table B.18) proposed by ter Horst [2005b] to extend RDFS.

owl:FunctionalProperty rdfs:subClassOf rdf:Property .	owl:equivalentProperty rdfs:range rdf:Property .
owl:InverseFunctionalProperty rdfs:subClassOf rdf:Property .	owl:Restriction rdfs:subClassOf rdfs:Class .
owl:SymmetricProperty rdfs:subClassOf rdf:Property .	owl:onProperty rdfs:domain owl:Restriction .
owl:TransitiveProperty rdfs:subClassOf rdf:Property .	owl:onProperty rdfs:range rdf:Property .
owl:sameAs rdf:type rdf:Property .	owl:hasValue rdfs:domain owl:Restriction .
owl:inverseOf rdf:type rdf:Property .	owl:someValuesFrom rdfs:domain owl:Restriction .
owl:inverseOf rdfs:domain rdf:Property .	owl:someValuesFrom rdfs:range rdfs:Class .
owl:inverseOf rdfs:range rdf:Property .	owl:allValuesFrom rdfs:domain owl:Restriction .
owl:equivalentClass rdf:type rdf:Property .	owl:allValuesFrom rdfs:range rdfs:Class .
owl:equivalentProperty rdf:type rdf:Property .	owl:differentFrom rdf:type rdf:Property .
owl:equivalentClass rdfs:domain rdfs:Class .	owl:disjointWith rdfs:domain rdfs:Class .
owl:equivalentProperty rdfs:domain rdf:Property .	owl:disjointWith rdfs:range rdfs:Class .

Table B.17: P-axiomatic triples [Hayes, 2004]

ID	**Body**		**Head**
	terminological	*assertional*	
rdfp1	**?p** a owl:FunctionalProperty .	?x ?p ?y , ?z .	?y owl:sameAs ?z .
rdfp2	**?p** a owl:InverseFunctionalProperty .	?x ?p ?z . ?y ?p ?z .	?x owl:sameAs ?y .
rdfp3	**?p** a owl:SymmetricProperty .	?x ?p ?y .	?y ?p ?x .
rdfp4	**?p** a owl:TransitiveProperty .	?x ?p ?y . ?y ?p ?z .	?x ?p ?z .
rdfp5a		?x ?p ?y .	?x owl:sameAs ?x .
rdfp5b		?x ?p ?y .	?y owl:sameAs ?y .
rdfp6		?x owl:sameAs ?y .	?y owl:sameAs ?x .
rdfp7		?x owl:sameAs ?y . ?y owl:sameAs ?z .	?x owl:sameAs ?z .
rdfp8a	**?p** owl:inverseOf ?q .	?x ?p ?y .	?y ?q ?x .
rdfp8b	?p owl:inverseOf **?q** .	?x ?q ?y .	?y ?p ?x .
rdfp9	**?c** a owl:Class .	?c owl:sameAs ?d .	?c rdfs:subClassOf ?d .
rdfp10	**?p** a owl:Property .	?p owl:sameAs ?q .	?p rdfs:subPropertyOf ?q .
rdfp11		?x owl:sameAs ?x′ . ?y owl:sameAs ?y′ . ?x ?p ?y .	?x′ ?p ?y′ .
rdfp12a	?c owl:equivalentClass ?d .		?c rdfs:subClassOf ?d .
rdfp12b	?c owl:equivalentClass ?d .		?d rdfs:subClassOf ?c .
rdfp12c	?c rdfs:subClassOf ?d . ?d rdfs:subClassOf ?c .		?c owl:equivalentClass ?d .
rdfp13a	?p owl:equivalentProperty ?q .		?p rdfs:subPropertyOf ?q .
rdfp13b	?p owl:equivalentProperty ?q .		?q rdfs:subPropertyOf ?p .
rdfp13c	?p rdfs:subPropertyOf ?q . ?q rdfs:subPropertyOf ?p .		?p owl:equivalentProperty ?q .
rdfp14a	?c owl:hasValue **?y** ; owl:onProperty **?p** .	?x ?p ?y .	?x a ?c .
rdfp14b	**?c** owl:hasValue ?y ; owl:onProperty ?p .	?x a ?c .	?x ?p ?y .
rdfp15	?c owl:someValuesFrom **?d** ; owl:onProperty **?p** .	?x ?p ?y . ?y a ?d .	?x a ?c .
rdfp16	**?c** owl:allValuesFrom ?d ; owl:onProperty **?p** .	?x a ?c; ?p ?y .	?y a ?d .
includes RDFS entailment and P-axiomatic triples			

Table B.18: P-entailment rules [ter Horst, 2005b]

Appendix C

Ranking Algorithms

Herein, we provide the detailed algorithms used for extracting, preparing and ranking the source level graph as used in Chapter 6. In particular, we provide the algorithms for parallel extraction and preparation of the sub-graphs on the slave machines: (i) extracting the source-level graph (Algorithm C.1); (ii) rewriting the graph with respect to redirect information (Algorithm C.2); (iii) pruning the graph with respect to the list of valid contexts (Algorithm C.3). Subsequently, the subgraphs are merge-sorted onto the master machine, which calculates the PageRank scores for the vertices (sources) in the graph as follows: (i) count the vertices and derive a list of dangling-nodes (Algorithm C.4); (ii) perform the power iteration algorithm to calculate the ranks (Algorithm C.5).

The algorithms are heavily based on on-disk operations: in the algorithms, we use `typewriter font` to denote on-disk operations and files. In particular, the algorithms are all based around sorting/scanning and *merge-joins*: a merge-join requires two or more lists of tuples to be sorted by a common join element, where the tuples can be iterated over in sorted order with the iterators kept "aligned" on the join element; we mark use of merge-joins in the algorithms using "m-join" in the comments.

Algorithm C.1: Extract raw sub-graph

```
Require: QUADS: Q                                   /* $\langle s,p,o,c \rangle_{0 \ldots n}$ sorted by c */
links ← {}, L ← {}
for all $\langle s,p,o,c \rangle_i \in$ Q do
    if $c_i \neq c_{i-1}$ then
        write(links, L)
        links ← {}
    end if
    for all $u \in \mathsf{U} \mid u \in \{s_i, p_i, o_i\} \wedge u \neq c_i$ do
        links ← links $\cup \{\langle c_i, u \rangle\}$
    end for
end for
write(links, L)
return L                                            /* unsorted on-disk outlinks */
```

Algorithm C.2: Rewrite graph wrt. redirects

```
Require: RAW LINKS: L                                    /* $\langle u,v\rangle_{0\ldots m}$ unsorted */
Require: REDIRECTS: R                         /* $\langle f,t\rangle_{0\ldots n}$ sorted by unique $f$ */
Require: MAX. ITERS.: $I$                                          /* typ. $I \leftarrow 5$ */
R$^-$ ← sortUnique$^-$(L)                                          /* sort by $v$ */
$i \leftarrow 0$; G$_\delta^-$ ← G$^-$
while G$_\delta^- \neq \emptyset \wedge i < I$ do
    $k \leftarrow 0$; G$_i^-$ ← {}; G$_{tmp}^-$ ← {}
    for all $\langle u,v\rangle_j \in$ G$_\delta^-$ do
        if $j = 0 \vee v_j \neq v_{j-1}$ then
            $rewrite \leftarrow \bot$
            if $\exists\langle f,t\rangle_k \in$ R $\mid f_k = v_j$ then                 /* m-join */
                $rewrite \leftarrow t_k$
            end if
        end if
        if $rewrite = \bot$ then
            write($\langle u,v\rangle_j$,G$_i^-$)
        else if $rewrite \neq u_j$ then
            write($\langle u_j, rewrite\rangle$,G$_{tmp}^-$)
        end if
    end for
    $i$++; G$_\delta^-$ ← G$_{tmp}^-$;
end while
G$_r^-$ ← mergeSortUnique({G$_0^-$, ..., G$_{i-1}^-$})
return G$_r^-$                              /* on-disk, rewritten, sorted inlinks */
```

Algorithm C.3: Prune graph by contexts

```
Require: NEW LINKS: G$_r^-$                            /* $\langle u,v\rangle_{0\ldots m}$ sorted by $v$ */
Require: CONTEXTS: C                                   /* $\langle c_1,\ldots,c_n\rangle$ sorted */
G$_p^-$ ← {}
for all $\langle u,v\rangle_i \in$ G$_r^-$ do
    if $i = 0 \vee c_i \neq c_{i-1}$ then
        $write \leftarrow$ false
        if $c_j \in$ C then                                          /* m-join */
            $write \leftarrow$ true
        end if
    end if
    if $write$ then
        write($\langle u,v\rangle_i$, G$_p^-$)
    end if
end for
return G$_p^-$                       /* on-disk, pruned, rewritten, sorted inlinks */
```

Algorithm C.4: Analyse graph

```
Require: OUT-LINKS: G                                    /* $\langle u,v\rangle_{0\ldots n}$ sorted by $u$ */
Require: IN-LINKS: $G^{-}$                               /* $\langle w,x\rangle_{0\ldots n}$ sorted by $x$ */
$V \leftarrow 0$                                         /* vertex count */
$u_{-1} \leftarrow \perp$
for all $\langle u,v\rangle_i \in G$ do
    if $i = 0 \vee u_i \neq u_{i-1}$ then
        $V{+}{+}$
        for all $\langle w,x\rangle_j \in G^{-} \mid u_{i-1} < x_j < u_i$ do    /* m-join */
            $V{+}{+}$; write($x_j$, DANGLE)
        end for
    end if
end for
for all $\langle w,x\rangle_j \in G^{-} \mid x_j > u_n$ do                      /* m-join */
    $V{+}{+}$; write($x_j$, DANGLE)
end for
return DANGLE                                /* sorted, on-disk list of dangling vertices */
return $V$                                   /* number of unique vertices */
```

Algorithm C.5: Rank graph

```
Require: OUT-LINKS: G                                  /* $\langle u,v\rangle_{0\ldots m}$ sorted by $u$ */
Require: DANGLING: DANGLE                              /* $\langle y_0,\ldots y_n\rangle$ sorted */
Require: MAX. ITERS.: $I$                               /* typ. $I \leftarrow 10$ */
Require: DAMPING FACTOR: $D$                            /* typ. $D \leftarrow 0.85$ */
Require: VERTEX COUNT: $V$
 1: $i \leftarrow 0$; $initial \leftarrow \frac{1}{V}$; $min \leftarrow \frac{1-D}{V}$
 2: $dangle \leftarrow D * initial * |$DANGLE$|$
 3:   /* GENERATE UNSORTED VERTEX/RANK PAIRS ... */
 4: while $i < I$ do
 5:    $min_i \leftarrow min + \frac{dangle}{V}$; PR$_{tmp} \leftarrow \{\}$;
 6:    for all $z_j \in$ DANGLE do                      /* $z_j$ has no outlinks */
 7:       write($\langle z_j, min_i\rangle$, PR$_{tmp}$)
 8:    end for
 9:    $out_j \leftarrow \{\}$; $rank \leftarrow initial$
10:    for all $\langle u,v\rangle_j \in$ G do          /* get ranks thru strong links */
11:       if $j \neq 0 \wedge u_j \neq u_{j-1}$ then
12:          write($\langle u_{j-1}, min_i\rangle$, PR$_{tmp}$)
13:          if $i \neq 0$ then
14:             $rank \leftarrow$ getRank($u_{j-1}$, PR$_i$)           /* m-join */
15:          end if
16:          for all $v_k \in out$ do
17:             write($\langle v_k, \frac{rank}{|out|}\rangle$, PR$_{tmp}$)
18:          end for
19:       end if
20:       $out_j \leftarrow out_j \cup \{v_j\}$
21:    end for
22:    do lines 12-18 for last $u_{j-1} \leftarrow u_m$
23:      /* SORT/AGGREGATE VERTEX/RANK PAIRS ... */
24:    PR$_{i+1} \leftarrow \{\}$; $dangle \leftarrow 0$
25:    for all $\langle z,r\rangle_j \in$ sort(PR$_{tmp}$) do
26:       if $j \neq 0 \wedge z_j \neq z_{j-1}$ then
27:          if $z_{j-1} \in$ DANGLE then                       /* m-join */
28:             $dangle \leftarrow dangle + rank$
29:          end if
30:          write($\langle z_{j-1}, rank\rangle$, PR$_{i+1}$)
31:       end if
32:       $rank \leftarrow rank + r_j$
33:    end for
34:    do lines 27-30 for last $z_{j-1} \leftarrow z_l$
35:    $i$++                                                /* iterate */
36: end while
37: return PR$_I$                           /* on-disk, sorted vertex/rank pairs */
```

Appendix D

Concurrence Analysis

In this chapter, we introduce methods for deriving a weighted concurrence score between entities in the Linked Data corpus: we define *entity concurrence* as the sharing of outlinks, inlinks and attribute values, denoting a specific form of similarity. We use these concurrence measures to materialise new links between such entities, and will also leverage the concurrence measures in § 7.6 for disambiguating entities. The methods described herein are based on preliminary works we presented in [Hogan et al., 2010d], where we:

- investigated domain-agnostic statistical methods for performing consolidation and identifying equivalent entities;
- formulated an initial small-scale (5.6 million triples) evaluation corpus for the statistical consolidation using reasoning consolidation as a best-effort "gold-standard".

The evaluation we presented in [Hogan et al., 2010d] provided mixed results, where we found some correlation between the reasoning consolidation and the statistical methods, but we also found that our methods gave incorrect results at high degrees of confidence for entities that were clearly not equivalent, but intuitively shared many links and attribute values in common. This of course highlights a crucial fallacy in our speculative approach: in almost all cases, even the highest degree of similarity/concurrence does not necessarily indicate equivalence or co-reference (cf. [Halpin et al., 2010b, § 4.4]). Similar philosophical issues arise with respect to handling transitivity for the weighted "equivalences" derived [Cock and Kerre, 2003; Klawonn, 2003].

However, deriving weighted concurrence measures has applications other than approximative consolidation: in particular, we can materialise named relationships between entities which share a lot in common, thus increasing the level of inter-linkage between entities in the corpus. Also, as we see in § 7.6, we can leverage the concurrence metrics to "rebuild" erroneous equivalence classes found during the disambiguation step. Thus, we present a modified version of the statistical analysis presented in [Hogan et al., 2010d], describe a scalable and distributed implementation thereof, and finally evaluate the approach with respect to finding highly-concurring entities in our 1 billion triple Linked Data corpus.

Note that we will apply our concurrence analysis over the consolidated corpus, as generated by the extended consolidation approach of § *7.4.*

D.1 High-level Approach

Our statistical concurrence analysis inherits similar *primary requirements* to that imposed for consolidation: the approach should be **scalable**, **fully automatic**, and **domain agnostic** to be applicable in our scenario.

Similarly, with respect to *secondary criteria*, the approach should be **efficient to compute**, should give **high precision**, and should give **high recall**. Compared to consolidation, high precision is not as critical for our statistical use-case: for example, taking SWSE as our use-case, we aim to use concurrency measures as a means of *suggesting* additional navigation steps for users browsing the entities—if the suggestion is uninteresting, it can be ignored, whereas incorrect consolidation will often lead to conspicuously garbled results, aggregating data on multiple disparate entities.

Thus, our requirements (particularly for scale) preclude the possibility of complex analyses or any form of pair-wise comparison, etc. Instead, we aim to design lightweight methods implementable by means of distributed sorts and scans over the corpus. Our methods are designed around the following intuitions and assumptions:

1. the concurrency of entities is measured as a function of their *shared pairs*, be they predicate-subject (loosely, inlinks), or predicate-object pairs (loosely, outlinks or attribute values);

2. the concurrence measure should give a *higher weight to exclusive shared-pairs*—pairs which are typically shared by few entities, for edges (predicates) which typically have a low in-degree/out-degree;

3. with the possible exception of correlated pairs—where pairs might not be independent—*each additional shared pair should increase the concurrency* of the entities: we assume that a shared pair *cannot* reduce the measured concurrency of the sharing entities;

4. *a small set of strongly exclusive property-pairs should be more influential than a large set of weakly exclusive pairs*: i.e., a few rarely-shared pairs should be rewarded a higher concurrence value than many frequently-shared pairs;

5. *correlation* may exist between shared pairs—e.g., two entities may share an inlink and an inverse-outlink to the same node (e.g., `foaf:depiction`, `foaf:depicts`), or may share a large number of shared pairs for a given property (e.g., two entities co-authoring one paper are more likely to co-author subsequent papers)—where we wish to dampen the cumulative effect of correlation in the concurrency analysis.

In fact, the concurrency analysis follows a similar principle to the consolidation presented in §§ 7.3 & 7.4, where instead of considering crisp functional and inverse-functional properties as given by the semantics of the data, we attempt to identify properties which are quasi-functional, quasi-inverse-functional, or what we more generally term *exclusive*: we determine the degree to which the values of properties (here abstracting directionality) are unique to an entity or set of entities.[1] The concurrency between two entities then becomes an aggregation of the weights for the property-value pairs they share in common.

To take a simple running example, consider the data in Listing D.1 where we want to determine the level of (relative) concurrency between three colleagues: `ex:Alice`, `ex:Bob` and `ex:Claire`: i.e., how much do they coincide/concur with respect to exclusive shared pairs.

D.1.1 Quantifying Concurrence

First, we want to characterise the uniqueness of properties; thus, we analyse their *observed* cardinality and inverse-cardinality as found in the corpus (in contrast to their defined cardinality as possibly given by the formal semantics):

Definition D.1 (Observed Cardinality) *Let G be an RDF graph, p be a property used as a predicate in G and s be a subject in G. The* observed cardinality *(or henceforth in this section, simply* cardinality*) of p*

[1] We note that a high exclusivity roughly corresponds to a high *selectivity*, and vice-versa.

Listing D.1: Running example for concurrence measures

```
dblp:AliceB10 foaf:maker ex:Alice .
dblp:AliceB10 foaf:maker ex:Bob .

ex:Alice foaf:gender "female" .
ex:Alice foaf:workplaceHomepage <http://deri.ie/> .

ex:Bob foaf:gender "male" .
ex:Bob foaf:workplaceHomepage <http://deri.ie/> .

ex:Claire foaf:gender "female" .
ex:Claire foaf:workplaceHomepage <http://deri.ie/> .
```

wrt s in G*, is given as follows:*

$$\mathsf{Card}_G(p,s) := |\{o \in \mathsf{C} : (s,p,o) \in G\}|\,.$$

Definition D.2 (Observed Inverse-Cardinality) *Let G and p be as before, and let o be an object in G. The* observed inverse-cardinality *(or henceforth in this chapter, simply* inverse-cardinality*)* of p wrt o in G*, is given as follows:*

$$\mathsf{ICard}_G(p,o) := |\{s \in \mathsf{U} \cup \mathsf{B} : (s,p,o) \in G\}|\,.$$

Thus, loosely, the observed cardinality of a property-subject pair is the number of unique objects it appears with in the graph (or unique triples it appears in); letting G_{ex} denote our example graph, then, e.g., $\mathsf{Card}_{G_{ex}}(\texttt{foaf:maker}, \texttt{dblp:AliceB10}) = 2$. We see this value as a good indicator of how *exclusive* (or *selective*) a given property-subject pair is, where sets of entities appearing in the object position of low-cardinality pairs are considered to *concur* more than those appearing with high-cardinality pairs. The observed inverse-cardinality of a property-object pair is the number of unique subjects it appears with in the graph—e.g., $\mathsf{ICard}_{G_{ex}}(\texttt{foaf:gender}, ''\texttt{female}'') = 2$. Both directions are considered analogous for deriving concurrence scores—note however that we do not consider concurrence for literals (i.e., we do not derive concurrence for literals which share a given predicate-subject pair; we do of course consider concurrence for subjects with the same literal value for a given predicate).

To avoid unnecessary duplication, we henceforth focus on describing only the inverse-cardinality statistics of a property, where the analogous metrics for plain-cardinality can be derived by switching subject and object (that is, switching directionality)—we choose the inverse direction as perhaps being more intuitive, indicating concurrence of entities in the subject position based on predicate-object pairs they share.

Definition D.3 (Average Inverse-Cardinality) *Let G be an RDF graph, and p be a property used as a predicate in G. The* average inverse-cardinality *of p, written $\mathsf{AIC}_G(p)$, is the average of the non-zero inverse-cardinalities of p in the graph G. Formally:*

$$\mathsf{AIC}_G(p) = \frac{|\{(s,o) \mid (s,p,o) \in G\}|}{|\{o \mid \exists s : (s,p,o) \in G\}|}$$

.

The average cardinality of a property is defined analogously. Note that the (inverse-)cardinality value of any term appearing as a predicate in the graph is necessarily greater-than or equal-to one: the numerator is by definition greater-than or equal-to the denominator.

Example D.1 *Referring back to the example graph in Listing D.1,* $\mathsf{AIC}_{G_{ex}}(\texttt{foaf:gender}) = 1.5$*, which can be viewed equivalently as the average non-zero cardinalities of* `foaf:gender` *(*1 *for* `"male"` *and* 2 *for* `"female"`*), or the number of triples with predicate* `foaf:gender` *divided by the number of unique values appearing in the object position of such triples (*$\frac{3}{2}$*).* ◇

Informally, we call a property p for which we observe $\mathsf{AIC}_G(p) \approx 1$, a *quasi-inverse-functional property* with respect to the graph G, and analogously properties for which we observe $\mathsf{AC}_G(p) \approx 1$ as *quasi-functional properties*. We see the values of such properties—in their respective directions—as being very exceptional: very rarely shared by entities. Thus, we would expect a property such as `foaf:gender` to have a high $\mathsf{AIC}_G(p)$ since there are (typically) only two object-values (`"male"`, `"female"`) shared by a large number of entities, whereas we would expect a property such as `foaf:workplaceHomepage` to have a lower $\mathsf{AIC}_G(p)$ since there are arbitrarily many values to be shared amongst the entities; given such an observation, we then surmise that a shared `foaf:gender` value represents a weaker "indicator" of concurrence than a shared value for `foaf:workplaceHomepage`.

Given that we deal with incomplete information under the Open World Assumption underlying RDF(S)/OWL, we also wish to weight the average (inverse) cardinality values for properties with a low number of observations towards a global mean—consider a fictional property `ex:maritalStatus` for which we only encounter a few predicate-usages in a given graph, and consider two entities given the value `"married"`: given sparse inverse-cardinality observations, we may naïvely over-estimate the significance of this property-object pair as an indicator for concurrence. Thus, we use a credibility formula as follows to weight properties with few observations towards a global mean (as per the intuition for the *cur* score in § 4.1.5:

Definition D.4 (Adjusted Average Inverse-Cardinality) *Let p be a property appearing as a predicate in the graph G. The* adjusted average cardinality of p with respect to G *is then*

$$\mathsf{AAIC}_G(p) = \frac{\mathsf{AIC}_G(p) \times |G^{\overrightarrow{p}}| + \overline{\mathrm{AIC}_G} \times |G^{\rightarrow}|}{|G^{\overrightarrow{p}}| + |G^{\rightarrow}|} \tag{D.1}$$

where $|G^{\overrightarrow{p}}|$ is the number of distinct objects that appear in a triple with p as a predicate (the denominator of Definition D.3), $\overline{\mathrm{AIC}_G}$ is the average inverse-cardinality for all predicate-object pairs (formally, $\overline{\mathrm{AIC}_G} = \frac{|G|}{|\{(p,o) | \exists s:(s,p,o) \in G\}|}$), and $|G^{\rightarrow}|$ is the average number of distinct objects for all predicates in the graph (formally, $|G^{\rightarrow}| = \frac{|\{(p,o) | \exists s:(s,p,o) \in G\}|}{|\{p | \exists s, \exists o:(s,p,o) \in G\}|}$)

Some reduction is possible, following $\mathsf{AIC}_G(p) \times |G^{\overrightarrow{p}}| = |\{(s,o) \mid (s,p,o) \in G\}|$ denoting the number of triples for which p appears as a predicate in graph G, and $\overline{\mathrm{AIC}_G} \times |G^{\rightarrow}| = \frac{|G|}{|\{p | \exists s, \exists o:(s,p,o) \in G\}|}$, denoting the average number of triples per predicate. We maintain Equation D.1 in the given unreduced form as it more clearly corresponds to the structure of a standard credibility formula: the reading ($\mathsf{AIC}_G(p)$) is dampened towards a mean ($\overline{\mathrm{AIC}_G}$) by a factor determined by the size of the sample used to derive the reading ($|G^{\overrightarrow{p}}|$) relative to the average sample size ($|G^{\rightarrow}|$).

Now, we move towards combining these metrics to determine the concurrency of entities who share a given non-empty set of property-value pairs. To do so, we combine the adjusted average (inverse) cardinality values which apply generically to properties, and the (inverse) cardinality values which apply to a given property-value pair. For example, take the property `foaf:workplaceHomepage`: entities that share a value referential to a large company—e.g., `http://google.com/`—should not gain as much concurrence as entities that share a value referential to a smaller company—e.g., `http://deri.ie/` (we will see this in Example D.2). Conversely, consider a fictional property `ex:citizenOf`—which relates a citizen to its country—for which we find many observations in our corpus, returning a high AAIC value, and consider that only two entities share the value `ex:Vanuatu` for this property: given that our data are incomplete, we can use the high AAIC value of `ex:citizenOf` to determine that the property is usually not exclusive, and that it is generally not a good indicator of concurrence.[2]

We start by assigning a coefficient to each pair (p, o) and each pair (p, s) that occur in the dataset, where the coefficient is an indicator of how exclusive that pair is:

Definition D.5 (Concurrence Coefficients) *The* concurrence-coefficient of a predicate-subject pair (p, s) *with respect to a graph G is given as:*

$$\mathsf{C}_G(p, s) = \frac{1}{\mathsf{Card}_G(p, s) \times \mathsf{AAC}_G(p)}$$

and the concurrence-coefficient of a predicate-object pair (p, o) *with respect to a graph G is analogously given as:*

$$\mathsf{IC}_G(p, o) = \frac{1}{\mathsf{ICard}_G(p, o) \times \mathsf{AAIC}_G(p)}$$

Again, please note that these coefficients fall into the interval $]0, 1]$ since the denominator, by definition, is necessarily greater than one.

Example D.2 *Let* p_{wh} = `foaf:workplaceHomepage` *and say that we compute* $\mathsf{AAIC}_G(p_{wh}) = 7$ *from a large number of observations, indicating that each workplace homepage in the graph G is linked to by, on average, seven employees. Further, let* o_g = `http://google.com/` *and assume that* o_g *occurs 2,000 times as a value for* p_{wh}*:* $\mathsf{ICard}_G(p_{wh}, o_g) = 2{,}000$*; now,* $\mathsf{IC}_G(p_{wh}, o_g) = \frac{1}{2{,}000 \times 7} = 0.00007$*. Also, let* o_d = `http://deri.ie/` *such that* $\mathsf{ICard}_G(p_{wh}, o_d) = 3$*; now,* $\mathsf{IC}_G(p_{wh}, o_d) = \frac{1}{100 \times 7} \approx 0.00143$*. Here, sharing DERI as a workplace will indicate a higher level of concurrence than analogously sharing Google (although both values are relatively low).* ◊

Finally, we require some means of aggregating the coefficients of the set of pairs that two entities share to derive the final concurrence measure.

Definition D.6 (Aggregated Concurrence Score) *Let* $Z = (z_1, \ldots z_n)$ *be a tuple such that for each* $i = 1, \ldots, n$, $z_i \in]0, 1]$*. The aggregated concurrence value* agg_n *is computed iteratively: starting with* $\mathsf{agg}_0 = 0$*, then for each* $k = 1 \ldots n$, $\mathsf{agg}_k = z_k + \mathsf{agg}_{k-1} - z_k \times \mathsf{agg}_{k-1}$.

The computation of the agg value is the same process as determining the probability of two independent events occurring—$P(A \vee B) = P(A) + P(B) - P(A \times B)$—which is by definition commutative and associative, and thus computation is independent of the order of the elements in the tuple. It may be more accurate to view the coefficients as *fuzzy values*, and the aggregation function as a disjunctive combination in some extensions of fuzzy logic [Zadeh et al., 1996].

Example D.3 *Let* $Z_a := (0.5, 0.3)$ *denote two concurrence coefficients for highly-exclusive pairs shared by two entities, where*

$$\mathsf{agg}(Z_a) = 0.5 + (1 - 0.5) \times 0.3 = 0.65\,.$$

Now, Let $Z_b := (0.3, 0.25, 0.15, 0.1)$ *denote a set of concurrence coefficients for three less exclusive pairs, where*

[2] Here, we try to distinguish between property-value pairs which are exclusive in reality (i.e., on the level of what's signified) and those which are exclusive in the given graph. Admittedly, one could think of counter-examples where not including the general statistics of the property may yield a better indication of weighted concurrence, particularly for generic properties which can be applied in many contexts; for example, consider the exclusive predicate-object pair (`dcterms:subject`, `category:Koenigsegg_vehicles`) given for a non-exclusive property.

$$\begin{aligned}
\mathsf{agg}_1(Z_b) &= 0 + (1-0) \times 0.3 &&= 0.3\\
\mathsf{agg}_2(Z_b) &= 0.3 + (1-0.3) \times 0.25 &&= 0.475\\
\mathsf{agg}_3(Z_b) &= 0.475 + (1-0.475) \times 0.15 &&= 0.55375\\
\mathsf{agg}(Z_b) = \mathsf{agg}_4(Z_b) &= 0.55375 + (1-0.55375) \times 0.1 &&= 0.598375
\end{aligned}$$

Although $\Sigma Z_a = \Sigma Z_b$, we see that the agg *gives a higher score to Z_a: although Z_a has fewer coefficients, it has stronger coefficients.* ◊

However, we acknowledge that the underlying coefficients may not be derived from strictly independent phenomena: there may indeed be correlation between the property-value pairs that two entities share. To illustrate, we reintroduce a relevant example from [Hogan et al., 2010d] shown in Figure D.1, where we see two researchers that have co-authored many papers together, have the same affiliation, and are based in the same country.

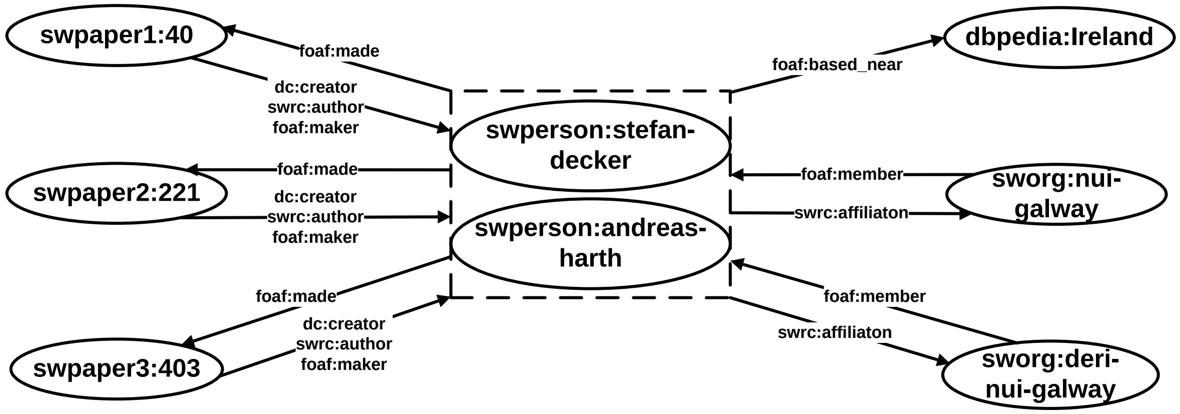


Figure D.1: Example of same-value, inter-property and intra-property correlation for shared inlink/outlink pairs, where the two entities under comparison are highlighted in the dashed box

This example illustrates three categories of concurrence correlation:

1. *same-value correlation* where two entities may be linked to the same value by multiple predicates in either direction (e.g., `foaf:made`, `dc:creator`, `swrc:author`, `foaf:maker`);
2. *intra-property correlation* where two entities which share a given property-value pair are likely to share further values for the same property (e.g., co-authors sharing one value for `foaf:made` are more likely to share further values);
3. *inter-property correlation* where two entities sharing a given property-value pair are likely to share further distinct but related property-value pairs (e.g., having the same value for `swrc:affiliation` and `foaf:based_near`).

Ideally, we would like to reflect such correlation in the computation of the concurrence between the two entities.

Regarding *same-value correlation*, for a value with multiple edges shared between two entities, we choose the shared predicate edge with the lowest AA[I]C value and disregard the other edges: i.e., we only consider the most exclusive property used by both entities to link to the given value and prune the other edges.

Regarding *intra-property correlation*, we apply a lower-level aggregation for each predicate in the set of shared predicate-value pairs. Instead of aggregating a single tuple of coefficients, we generate a bag of tuples

$\mathbf{Z} = \{Z_{p_1}, \ldots, Z_{p_n}, Z_{p'_1}, \ldots, Z_{p'_n}\}$, where each element Z_{p_i} represents the tuple of (non-pruned) coefficients generated for inlinks by the predicate p_i, and where each element $Z_{p'_i}$ represents the coefficients generated for outlinks with the predicate p_i.[3]

We then aggregate this bag as follows:

$$\overline{\mathsf{agg}}(\mathbf{Z}) = \mathsf{agg}\left(\frac{\mathsf{agg}(Z_{p_1})}{\mathsf{AAC}(p_1)}, \ldots, \frac{\mathsf{agg}(Z_{p_n})}{\mathsf{AAC}(p_n)}, \frac{\mathsf{agg}(Z_{p'_1})}{\mathsf{AAIC}(p_1)}, \ldots, \frac{\mathsf{agg}(Z_{p'_n})}{\mathsf{AAIC}(p_n)}\right)$$

Thus, the total contribution possible through a given predicate (e.g., `foaf:made`) has an upper-bound set as its $\frac{1}{\mathsf{AA[I]C}}$ value, where each successive shared value for that predicate (e.g., each successive co-authored paper) contributes positively (but increasingly less) to the overall concurrence measure. We illustrate with an example:

Example D.4 *Assume that we are deriving the concurrence of the two entities depicted in Figure D.1, and (for brevity) that we have knowledge of the edges on the right hand side of the figure; i.e.:*

1.	(`swperson:`x, `foaf:based_near`, `dbpedia:Ireland`)	0.00002
2.	(`swperson:`x, `swrc:affiliation`, `sworg:nui-galway`)	0.001
3.	(`sworg:nui-galway`, `foaf:member`, `swperson:`x)	0.003
4.	(`swperson:`x, `swrc:affiliation`, `sworg:deri-nui-galway`)	0.02
5.	(`sworg:deri-nui-galway`, `foaf:member`, `swperson:`x)	0.05

where `swperson:`x *refers to both* `swperson:`***`andreas-harth`*** *and* `swperson:`***`stefan-decker`***, *and where the value on the right hand side denotes the concurrence coefficient associated with that edge. First, we prune edges* (2) *and* (4) *since edge* (3) *represents a stronger edge to* `sworg:nui-galway` *and* (5) *represents a stronger edge to* `sworg:deri-nui-galway`.

1.	(`swperson:`x, `foaf:based_near`, `dbpedia:Ireland`)	0.00002
3.	(`sworg:nui-galway`, `foaf:member`, `swperson:`x)	0.003
5.	(`sworg:deri-nui-galway`, `foaf:member`, `swperson:`x)	0.05

Further, let p_a *denote* `foaf:based_near` *and* p_b *denote* `foaf:member`*; now we have:*

$$\mathbf{Z} := \{Z_{p_a}, Z_{p_b}, Z'_{p_a}, Z'_{p_b}\} = \{(0), (0.003, 0.05), (0.00002), (0)\}$$

now, letting $\mathsf{AAIC}(p_a) = 1,000$, $\mathsf{AAC}(p_b) = 10$, *we have:*

$$\begin{aligned}\overline{\mathsf{agg}}(\mathbf{Z}) &= \mathsf{agg}\left(0, \frac{\mathsf{agg}\big((0.003, 0.05)\big)}{10}, \frac{\mathsf{agg}\big((0.00002)\big)}{1,000}, 0\right)\\ &= \mathsf{agg}\big((0, 0.005285, 0.0000002, 0)\big)\\ &= \sim 0.005285018943\end{aligned}$$

◊

Finally, we note three issues with this approach:

- the absolute values are significantly reduced due to the additional $\mathsf{AA[I]C}$ denominators in the $\overline{\mathsf{agg}}$ calculation—however, absolute values are not important in our scenario, where we are more interested in relative values for comparison;

[3] For brevity, we omit the graph subscript.

Algorithm D.1: Computing AAC/AAIC values

```
Require: prp-ifp-INPUT: IN                                   /* on-disk input triples */
 1: sort IN by lexiographical ($s$–$p$–$o$) order to $\mathtt{IN}_s^+$
 2: $O := \{\}$; $i := 0$;
 3: $dist_o := \{\}$        /* a distrib. of obj. counts for each pred.; e.g., $p_x \mapsto \{(1,700),(2,321),\ldots\}$ */
 4: for all $t_i \in \mathtt{IN}_s^+$ do
 5:   if $i \neq 0 \wedge (t_i.s \neq t_{i-1}.s \vee t_i.p \neq t_{i-1}.p)$ then
 6:     $dist_o(t_{i-1}.p)(|O|)$++
 7:     $O := \{\}$
 8:   end if
 9:   $O := O \cup \{t_i.o\}$
10:   $i$++
11: end for
12: repeat Line 6 for final $t.p, O$
13: compute AAC() values from $dist_o$              /* as per Definition D.4 (switching direction) */
14: sort IN by inverse ($o$–$p$–$s$) order to $\mathtt{IN}_s^-$
15: $S := \{\}$; $i := 0$;
16: $dist_s := \{\}$                                /* a distrib. of subj. counts for each pred. */
17: for all $t_i \in \mathtt{IN}_s^-$ do
18:   if $i \neq 0 \wedge (t_i.o \neq t_{i-1}.o \vee t_i.p \neq t_{i-1}.p)$ then
19:     $dist_s(t_{i-1}.p)(|S|)$++
20:     $S := \{\}$
21:   end if
22:   $S := S \cup \{t_i.s\}$
23:   $i$++
24: end for
25: repeat Line 19 for final $t.p, S$
26: compute AAIC() values from $dist_s$                              /* as per Definition D.4 */
```

- we may prune correlated edges with *different* predicates, which may affect the $\overline{\mathsf{agg}}$ result in unintuitive ways: taking the previous example, if (3) and (4) were pruned instead, then edges (2) and (5) would have different predicates, and would not be aggregated together in the same manner as for (3) and (5) in the example, leading to a higher $\overline{\mathsf{agg}}$ result;

- along similar lines, our pruning and $\overline{\mathsf{agg}}$ derivation do not detect or counter-act *inter-property correlation*, which is perhaps a more difficult issue to address.

Having acknowledged the latter two weaknesses of our approach, we leave these issues open.

D.1.2 Implementing Entity-concurrence Analysis

We aim to implement the above methods using sorts and scans, and wish to avoid any form of complex indexing, or pair-wise comparison. Given that there are 23 thousand unique predicates found in the input corpus, we assume that we can fit the list of predicates and their associated statistics in memory—if such were not the case, one could consider an on-disk map with an in-memory LRU cache, where we would expect a high cache hit-rate based on the distribution of property occurrences in the data (cf. § 4.2).

Firstly, we wish to extract the statistics relating to the (inverse-)cardinalities of the predicates in the data; this process is outlined in Algorithm D.1 for reference. We first sort the data according to natural order (s, p, o), and then scan the data, computing the cardinality (number of distinct objects) for each (s, p) pair, and maintaining the distribution of object counts for each p found. For inverse-cardinality scores, we apply the same process, sorting instead by (o, p, s) order, counting the number of distinct subjects for each

Algorithm D.2: Computing concurrence values

```
Require: INPUT (LEX. ORDER): $\texttt{IN}_s^+$          /* on-disk triples sorted by s−p−o (from Alg. D.1) */
Require: INPUT (INV. ORDER): $\texttt{IN}_s^-$          /* on-disk triples sorted by o−p−s (from Alg. D.1) */
Require: AAC/AAIC MAPS: AAC/AAIC          /* in-memory: $p_x \mapsto \mathsf{AA[I]C}(p_x)$ (from Alg. D.1) */
Require: CONCURRENCE OUTPUT: CON_OUT          /* on-disk output */
 1: $O_{ps} := \{\}$; $i := 0$; CON_OUT_TMP $:= \{\}$
 2: for all $t_i \in \texttt{IN}_s^+$ do
 3:    if $i \neq 0 \wedge (t_i.s \neq t_{i-1}.s \vee t_i.p \neq t_{i-1}.p)$ then
 4:       compute $\mathsf{C}_G(p,s)$ from AAC and $|O_{ps}|$          /* as per Definition D.5 */
 5:       for all $(o_i, o_j) : o_i, o_j \in (O_{ps} \setminus \mathsf{L}), o_i <_c o_j$ do
 6:          write $(o_i, o_j, \mathsf{C}_G(p,s), p, s, -)$ to CON_OUT_TMP
 7:       end for
 8:    end if
 9:    $O_{ps} := O_{ps} \cup \{t_i.o\}$
10:    $i{+}{+}$
11: end for
12: repeat Lines 4–7 for final $t, O_{ps}$
13: $S_{po} := \{\}$; $i := 0$;
14: for all $t_i \in \texttt{IN}_s^-$ do
15:    if $i \neq 0 \wedge (t_i.o \neq t_{i-1}.o \vee t_i.p \neq t_{i-1}.p)$ then
16:       compute $\mathsf{IC}_G(p,o)$ from AAIC and $|S_{po}|$          /* as per Definition D.5 */
17:       for all $(s_i, s_j) : s_i, s_j \in S_{po}, s_i <_c s_j$ do
18:          write $(s_i, s_j, \mathsf{IC}_G(p,o), p, o, +)$ to CON_OUT_TMP
19:       end for
20:    end if
21:    $S_{po} := S_{po} \cup \{t_i.s\}$
22:    $i{+}{+}$
23: end for
24: repeat Lines 16–19 for final $t, S_{po}$
25:          /* CON_OUT_TMP contains tuples of the form $(e_a, e_b, c, p, v, \pm)$: compared entities $(e_a, e_b)$ , edge
    coefficient c, and edge (p, v, ±) [predicate/value/direction] */
26: $Edges := \{\}$
27: for all $tup_i \in$ CON_OUT_TMP do
28:    if $i \neq 0 \wedge (tup_i.e_a \neq tup_{i-1}.e_a \vee tup_i.e_b \neq tup_{i-1}.e_b)$ then
29:       create $\mathbf{Z}$ from $Edges$          /* as per Example D.4 */
30:       compute $\overline{\mathsf{agg}}(\mathbf{Z})$ using AAC/AAIC          /* as per Example D.4 */
31:       write $(tup_{i-1}.e_a, tup_{i-1}.e_b, \overline{\mathsf{agg}}(\mathbf{Z}))$ to CON_OUT
32:       write $(tup_{i-1}.e_b, tup_{i-1}.e_a, \overline{\mathsf{agg}}(\mathbf{Z}))$ to CON_OUT
33:    end if
34:    $Edges := Edges \cup \{(tup_i.c, tup_i.p, tup_i.v, tup_i.\pm)\}$
35:    $i{+}{+}$
36: end for
37: repeat Lines 29–32 for final $e_a, e_b, Edges$
```

(p,o) pair, and maintaining the distribution of subject counts for each p. After each scan, the statistics of the properties are adjusted according to the credibility formula in Definition D.4.

Thereafter, the process for generating the concurrence values is outlined in Algorithm D.2.

First, we scan the data sorted in lexicographical order, and for each (s,p) pair, for each set of objects O_{ps} found thereon, and for each pair in

$$\{(o_i, o_j) \mid o_i, o_j \in (O_{ps} \setminus \mathsf{L}), o_i <_c o_j\}$$

(where $<_c$ denotes the canonical order of Definition 7.1) we output the following sextuple to an on-disk file:

$$(o_i, o_j, \mathsf{C}(p,s), p, s, -)$$

where $\mathsf{C}(p,s) = \frac{1}{|O_{ps}| \times \mathsf{AAC}(p)}$. We apply the same process for the other direction, outputting analogous sextuples of the form:

$$(s_i, s_j, \mathsf{IC}(p,o), p, o, +)$$

We call the sets O_{ps} are their analogues S_{po} *concurrence classes*, denoting sets of entities which share the given predicate-subject/predicate-object pair respectively. Here, note that the '+' and '−' elements simply demarcate and track the directionality from which the tuple was generated, required for the final aggregation of the co-efficient scores. Similarly, we do not immediately materialise the symmetric concurrence scores, where we instead do so at the end so as to forego duplication of intermediary processing.

Once generated, we can sort the two files of tuples by their natural order, and perform a merge-join on the first two elements—generalising the directional o_i/s_i to simply e_i, each (e_i, e_j) pair denotes two entities which share some predicate-value pairs in common, where we can scan the sorted data and aggregate the final concurrence measure for each (e_i, e_j) pair using the information encoded in the respective tuples. We can thus generate (trivially sorted) tuples of the form (e_i, e_j, s), where s denotes the final aggregated concurrence score computed for the two entities; optionally, we can also write the symmetric concurrence tuples (e_j, e_i, s) which can be sorted separately as required.

Note that the number of tuples generated is quadratic with respect to the size of the respective concurrence class, which becomes a major impediment for scalability given the presence of large such sets—for example, consider a corpus containing 1 million persons sharing the value `"female"` for the property `foaf:gender`, where we would have to generate $\frac{10^{6\times 2} - 10^6}{2} \approx 500$ billion non-reflexive, non-symmetric concurrence tuples. However, we can leverage the fact that such sets can only invoke a minor influence on the final concurrence of their elements, given that the magnitude of the set—e.g., $|S_{po}|$—is a factor in the denominator of the computed $\mathsf{C}(p,o)$ score, such that $\mathsf{C}(p,o) \propto \frac{1}{|S_{op}|}$. Thus, in practice, we implement a maximum-size threshold for the S_{po} and O_{ps} concurrence classes: this threshold is selected based on a practical upper limit for raw similarity tuples to be generated, where the appropriate maximum class size can trivially be determined alongside the derivation of the predicate statistics. For the purpose of evaluation, we choose to keep the number of raw tuples generated at around ∼1 billion, and so set the maximum concurrence class size at 38: we arrive at this latter figure using an empirical analysis of the data presented in § D.4.

D.2 Distributed Implementation

Given the previous discussion, our distributed implementation is fairly straightforward as follows:

1. **coordinate**: the slave machines split their segment of the corpus according to a modulo-hash function on the subject position of the data, sort the segments, and send the split segments to the peer determined by the hash-function; the slaves simultaneously gather incoming sorted segments, and subsequently perform a merge-sort of the segments;

2. **coordinate**: the slave machines apply the same operation, this time hashing on object—triples with `rdf:type` as predicate are not included in the object-hashing; subsequently the slaves merge-sort the segments ordered by object;

3. **run**: the slave machines then extract predicate-level statistics, and statistics relating to the concurrence-class-size distribution which are used to decide upon the class size threshold;

4. **gather/flood/run**: the master machine gathers and aggregates the high-level statistics generated by the slave machines in the previous step and sends a copy of the global statistics back to each machine; the slaves subsequently generate the raw concurrence-encoding sextuples (as described in the previous section) from a scan of the data in both orders;
5. **coordinate**: the slave machines coordinate the locally generated sextuples according to the first element (join position) as before;
6. **run**: the slave machines aggregate the sextuples coordinated in the previous step, and produce the final non-symmetric concurrence tuples;
7. **run**: the slave machines produce the symmetric version of the concurrence tuples, and coordinate and sort on the first element.

Here, we make heavy use of the **coordinate** function to align data according to the join position required for the subsequent processing step—in particular, aligning the raw data by subject and object, and then the concurrence tuples analogously.

Note that we do not hash on the object position of `rdf:type` triples: our raw corpus contains 206.8 million such triples, and given the distribution of class memberships, we assume that hashing these values will lead to uneven distribution of data, and subsequently uneven load balancing—e.g., 79.2% of all class memberships are for `foaf:Person`, hashing on which would send 163.7 million triples to one machine, which alone is greater than the average number of triples we would expect per machine (139.8 million). In any case, given that our corpus contains 105 thousand unique values for `rdf:type`, we would expect the average-inverse-cardinality to be approximately 1,970—even for classes with two members, the potential effect on concurrence is negligible.

D.3 Performance Evaluation

We apply our concurrence analysis over the consolidated corpus derived in § 7.4. The total time taken was 13.9 h. Sorting, splitting and scattering the data according to subject on the slave machines took 3.06 h, with an average idle time of 7.7 min (4.2%). Subsequently merge-sorting the sorted segments took 1.13 h, with an average idle time of 5.4 min (8%). Analogously sorting, splitting and scattering the non-`rdf:type` statements by object took 2.93 h, with an average idle time of 11.8 min (6.7%). Merge sorting the data by object took 0.99 h, with an average idle time of 3.8 min (6.3%). Extracting the predicate statistics and threshold information from data sorted in both orders took 29 min, with an average idle time of 0.6 min (2.1%). Generating the raw, unsorted similarity tuples took 69.8 min with an average idle time of 2.1 min (3%). Sorting and coordinating the raw similarity tuples across the machines took 180.1 min, with an average idle time of 14.1 min (7.8%). Aggregating the final similarity took 67.8 min, with an average idle time of 1.2 min (1.8%).

Table D.1 presents a breakdown of the timing of the task. Although this task requires some aggregation of global knowledge by the master machine, the volume of data involved is minimal: a total of 2.1 minutes is spent on the master machine performing various minor tasks (initialisation, remote calls, logging, aggregation and broadcast of statistics). Thus, 99.7% of the task is performed in parallel on the slave machine. Although there is less time spent waiting for the master machine compared to the previous two tasks, deriving the concurrence measures involves three expensive sort/coordinate/merge-sort operations to redistribute and sort the data over the slave swarm. The slave machines were idle for, on average, 5.8% of the total task time; most of this idle time (99.6%) was spent waiting for peers. We note that the master machine was idle for almost the entire task (99.7%).

Category	min	% Total
Total execution time	835.4	100
Master (Local)		
Executing	2.1	0.3
Miscellaneous	2.1	0.3
Idle (waiting for slaves)	833.3	99.7
Slave (Parallel)		
Avg. Executing (total exc. idle)	786.6	94.2
Split/sort/scatter (subject)	175.9	21.1
Merge-sort (subject)	62.4	7.5
Split/sort/scatter (object)	164	19.6
Merge-sort (object)	55.6	6.6
Extract High-level Statistics	28.4	3.3
Generate Raw Concurrence Tuples	67.7	8.1
Cooordinate/Sort Concurrence Tuples	166	19.9
Merge-sort/Aggregate Similarity	66.6	8
Avg. Idle	48.8	5.8
Waiting for peers	46.7	5.6
Waiting for master	2.1	0.3

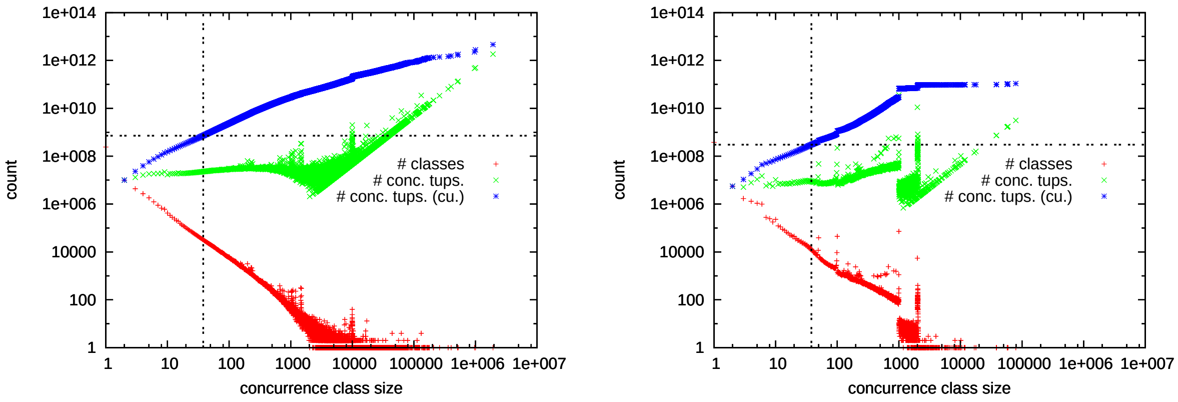
Table D.1: Breakdown of timing of distributed concurrence analysis

D.4 Results Evaluation

With respect to data distribution, after hashing on subject we observed an average absolute deviation (average distance from the mean) of 176 thousand triples across the slave machines, representing an average 0.13% deviation from the mean: near-optimal data distribution. After hashing on the object of non-`rdf:type` triples, we observed an average absolute deviation of 1.29 million triples across the machines, representing an average 1.1% deviation from the mean; in particular, we note that one machine was assigned 3.7 million triples above the mean (an additional 3.3% above the mean). Although not optimal, the percentage of data deviation given by hashing on object is still within the natural variation in run-times we have seen for the slave machines during most parallel tasks.

First, we empirically motivate our cut-off for the maximum equivalence class size we allow; for example, generating all pairwise concurrence tuples between the subjects which share the predicate-object edge (`rdf:type`, `foaf:Person`) would be completely infeasible, and where the concurrence coefficients would in any case have negligible value (see § D.1.2). Along these lines, in Figures D.2(a) and D.2(b), we illustrate the effect of including increasingly large concurrence classes on the number of raw concurrence tuples generated. Note that the count of concurrence class size reflects the number of edges that were attached to the given number of entities: for example, for predicate-object pairs, a concurrence class size of two reflects the number of predicate-object edges which had two subjects. Thereafter, the number of (non-reflexive, non-symmetric) concurrence tuples generated for each class size is calculated as $tup_x = c_x \times \frac{x^2+x}{2}$, where x denotes the concurrence class size, and where c_x denotes the count of classes of that size. Next, the cumulative count of concurrence tuples is given as $\widehat{tup_x} = \sum_{i \leq x} tup_i$, giving the number of tuples required to represent all concurrence classes up to that size. Finally, we show our cut-off ($max = 38$) intended to keep the total number of concurrence tuples at ∼1 billion: i.e., max is chosen as the lowest possible value for x such that $\widehat{tup_x^{po}} + \widehat{tup_x^{sp}} > 10^9$ holds, where $\widehat{tup_x^{po}}/\widehat{tup_x^{ps}}$ denotes the cumulative count (up to x) for predicate-object/predicate-subject concurrence classes respectively. With $max = 38$, we measure $\widehat{tup_{max}^{po}}$ to give 721 million concurrence tuples, and $\widehat{tup^{ps}_{max}}$ to give 303 million such tuples.

For the predicate-object pairs, we observe an apparent power-law relationship between the size of the concurrence class and the number of such classes observed. Second, we observe that the number of concurrences generated for each increasing concurrence class size initially remains fairly static—i.e., larger concurrence class sizes give quadratically more concurrences, but occur polynomially less often—until the point where the largest classes which generally only have one occurrence is reached, and the number of concurrences begins to increase quadratically. Also shown is the cumulative count of concurrence tuples generated for increasing class sizes, where we initially see rapid growth, which subsequently begins to flatten as the larger concurrence classes become more sparse (although more massive).



(a) Predicate-object pairs

(b) Predicate-subject pairs

Figure D.2: For predicate-subject edges and predicate-object edges (resp.), and for increasing sizes of generated concurrence classes, we show [in log/log] the count of (i) concurrence classes at that size [`#classes`]; (ii) concurrence tuples needed to represent all class *at that* size [`conc. tups.`]; and (iii) concurrence tuples needed to represent all classes *up to* that size [`#conc. tups. (cu.)`]; finally, we also show the concurrence class-size cut-off we implement to keep the *total* number of concurrence tuples at ∼ 1 billion [*dotted line*]

For the predicate-subject pairs, the same roughly holds true, although we see fewer of the very largest concurrence classes: the largest concurrence class given by a predicate-subject pair was 79 thousand, versus 1.9 million for the largest predicate-object pair, respectively given by the pairs (`kwa:map`, `macs:manual_rameau_lcsh`) and (`opiumfield:rating`, `""`). Also, we observe some "noise" where for milestone concurrence class sizes (esp., at 50, 100, 1,000, 2,000) we observe an unusual amount of classes. For example, there were 72 thousand concurrence classes of precisely size 1,000 (versus 88 concurrence classes at size 996)—the 1,000 limit was due to a FOAF exporter from the `hi5.com` which seemingly enforces that limit on the total "friends count" of users, translating into many users with precisely 1,000 values for `foaf:knows`.[4] Also for example, there were 5.5 thousand classes of size 2,000 (versus 6 classes of size 1,999)—almost all of these were due to an exporter from the `bio2rdf.org` domain which puts this limit on values for the `bio2rdf:linkedToFrom` property.[5] We also encountered unusually large numbers of classes approximating these milestones, such as 73 at 2,001. Such phenomena explain the staggered "spikes"and "discontinuities" in Figure D.2(b), which can be observed to correlate with such milestone values (in fact, similar but less noticeable spikes are also present in Figure D.2(a)).

With respect to the statistics of predicates, for the predicate-subject pairs, each predicate had an average

[4] cf. `http://api.hi5.com/rest/profile/foaf/100614697`; retr. 2011/09/22

[5] cf. `http://bio2rdf.org/mesh:D000123Q000235`; retr. 2011/09/22

of 25,229 unique objects for 37,953 total triples, giving an average cardinality of ∼1.5. We give the five predicates observed to have the lowest adjusted average cardinality in Table D.2; note that two of these properties will not generate any concurrences since they are perfectly unique to a given object (they have the same number of objects and triples), and that the values shown for AAC are approximate. For the predicate-object pairs, there was an average of 11,572 subjects for 20,532 triples, giving an average inverse-cardinality of ∼2.64; We give the five predicates observed to have the lowest adjusted average inverse cardinality in Table D.3; again, four of these properties will not generate any concurrences since they are perfectly unique to a given subject (they have the same number of subjects and triples), and again values shown for AAIC are approximate..

#	Predicate	Objects	Triples	∼AAC
1	`foaf:nick`	150,433,035	150,437,864	1.000
2	`lldpubmed:journal`	6,790,426	6,795,285	1.003
3	`rdf:value`	2,802,998	2,803,021	1.005
4	`eurostat:geo`	2,642,034	2,642,034	1.005
5	`eurostat:time`	2,641,532	2,641,532	1.005

Table D.2: Top five predicates with respect to lowest adjusted average cardinality (AAC)

#	Predicate	Subjects	Triples	∼AAIC
1	`lldpubmed:meshHeading`	2,121,384	2,121,384	1.009
2	`opiumfield:recommendation`	1,920,992	1,920,992	1.010
3	`fb:type.object.key`	1,108,187	1,108,187	1.017
4	`foaf:page`	1,702,704	1,712,970	1.017
5	`skipinions:hasFeature`	1,010,604	1,010,604	1.019

Table D.3: Top five predicates with respect to lowest adjusted average inverse-cardinality (AAIC)

Aggregation produced a final total of 636.9 million weighted concurrence pairs, with a mean concurrence weight of ∼0.0159. Of these pairs, 19.5 million involved a pair of identifiers from different PLDs (3.1%), whereas 617.4 million involved identifiers from the same PLD; however, the average confidence value for an intra-PLD pair was 0.446, versus 0.002 for inter-PLD pairs—although fewer intra-PLD concurrences are found, they typically have higher confidences.[6]

#	Entity Label 1	Entity Label 2	Shar. Edges
1	`New York City`	`New York State`	791
2	`London`	`England`	894
3	`Tokyo`	`Japan`	900
4	`Toronto`	`Ontario`	418
5	`Philadelphia`	`Pennsylvania`	217

Table D.4: Top five concurrent entities and the number of edges they share

In Table D.4, we give the labels of top five most concurrent entities, including the number of pairs they share—the confidence score for each of these pairs was >0.9999999. We note that they are all locations, where particularly on WIKIPEDIA (and thus filtering through to DBpedia), properties with location val-

[6] Note that we apply this analysis over the consolidated data, and thus this is an approximative reading for the purposes of illustration: we extract the PLDs from canonical identifiers, which are choosen based on arbitrary lexical ordering.

ues are typically duplicated (e.g., dbp:deathPlace, dbp:birthPlace, dbp:headquarters—properties that are *quasi-functional*); for example, New York City and New York State are both the dbp:deathPlace of dbpedia:Isacc_Asimov, etc.

#	**Ranked Entity**	**#Con.**	**"Closest" Entity**	**Val.**
1	Tim Berners-Lee	908	Lalana Kagal	0.83
2	Dan Brickley	2,552	Libby Miller	0.94
3	update.status.net	11	socialnetwork.ro	0.45
4	FOAF-a-matic	21	foaf.me	0.23
5	Evan Prodromou	3,367	Stav Prodromou	0.89

Table D.5: Breakdown of concurrences for top five ranked entities, ordered by rank, with, respectively, entity label, number of concurrent entities found, the label of the concurrent entity with the largest degree, and finally the degree value

In Table D.5, we give a description of the top concurrent entities found for the top-five ranked entities in our corpus—for brevity, again we show entity labels. In particular, we note that a large amount of concurrent entities are identified for the highly-ranked persons. With respect to the strongest concurrences: (i) Tim and his former student Lalana share twelve primarily academic links, coauthoring six papers; (ii) Dan and Libby, co-founders of the FOAF project, share 87 links, primarily 73 foaf:knows relations to and from the same people, as well as a co-authored paper, occupying the same professional positions, etc.;[7] (iii) update.status.net and socialnetwork.ro share a single foaf:accountServiceHomepage link from a common user; (iv) similarly, the FOAF-a-matic and foaf.me services share a single mvcb:generatorAgent inlink; (v) finally, Evan and Stav share 69 foaf:knows inlinks and outlinks exported from the identi.ca service.

We note that none of these prominent/high-confidence results indicate coreference.

[7] Notably, Leigh Dodds (creator of the FOAF-a-matic service) is linked by the property quaffing:drankBeerWith to both.

List of Algorithms

List of Figures

List of Tables

Listings

"I have made this letter longer than usual, only because I have not had the time to make it shorter."

—Blaise Pascal